STRUCTURED SYSTEMS ANALYSIS AND DESIGN

25

APTECH

STRUCTURED SYSTEMS ANALYSIS AND DESIGN

Structured Systems Analysis and Design

Aptech,
54-A, Sir M. Vasanji Road,
Andheri(E),
Bombay - 400 093.

Printed in India

1st Edition - printed in July 1995

Preface

Structured Systems Analysis and Design

Computer information systems are the heart of daily activities and a major consideration in corporate decision making. Just as knowledge of computers and computer programming has become a basic skill needed to survive in today's information-based society, so has the design of business systems. The development of information systems involves both systems analysts and those who will use the applications that emerge i.e. end-users. The analysis and design of computer systems involve many parts of the organisation and are not limited to the domain of computer specialists.

The objective of this book is to involve students in the ***systems thinking*** activity that is essential for the design and development of business systems. The chapters in the book guide you through the systems development life cycle and explain the techniques used in each phase. After a brief introduction to structured analysis and design techniques in the first chapter, the rest of the chapters explain how to use each of these structured techniques. Although various methodologies are available for analysis and design, this book focuses on structured techniques which is one of the widely used methodologies.

It is our continuos endeavour to bring to you the latest, the best and the most relevant matter related to computers.

The material in this book is brought to you by the Design team. The process of design has been part of the ISO 9001 certification for Aptech - IT division. Education Support Services. As part of Aptech's quality drive, this team is continuously researching and improving upon the curriculum to keep it in line with the industry trends.

This research and development team will be glad to receive any suggestions. The students must feel free to send feedback to the Design Head at Aptech's Head Office, Bombay.

Design Team

Table of Contents

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

Chapter 1

System Concepts

At the end of this session, you will be able to :

➔ *Understand what a system is*

➔ *Define categories of information systems*

➔ *Identify the various phases of the systems development life cycle*

➔ *Write a feasibility report for a system*

➔ *Know the various fact finding techniques*

➔ *Know the job of a systems analyst and the qualities of an analyst*

1.0 Introduction

We use the computer technology in many ways, from visible to invisible, spectacular to routine, video games and special effects for films and television to microwave ovens, electronic cameras, and automobile ignition systems. In business too, computers and information systems occupy a special place. Computers make possible the smooth and efficient operation of airline reservations, hospital record departments, accounting and payroll functions, electronic banking, telephone switching systems and countless other operations both large and small.

How do such complex information systems come into existence? Obviously through people. Technology has developed at a very fast rate, but the most important aspect of any system is the human know-how and the use of ideas to gear the computer so that it performs the required task. This process is essentially what systems development is all about.

To be of any use, a computer-based information system must function properly, be easy to use, and suit the organisation for which it has been designed. If it helps people do their jobs better and more efficiently, they will use it. If it is not helpful, they will surely avoid it.

1.1 What is a system

Systems are, in fact, all around us. For example you experience physical sensations by means of a complex nervous system which consists of parts, including the brain, spinal cord, nerves, and special sensitive cells that work together to make a person feel hot, cold etc.

A person communicates by means of a language, a highly developed system of words and symbols that convey meaning to each other . We all live according to an economic system in which goods and services are exchanged for other goods and services of comparable value and by which the participants to the exchange are benefited.

A system is simply a set of components that interact to accomplish some purpose.

Or

A system is an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective.

The components that make up systems may actually be other smaller systems; that is, systems may be made up of levels of systems, or subsystems.

Organisations consist of many business systems, each having the features of the general system.

The purposes of information systems are to process input, maintain files of data about the organisation, and produce information, reports, and other output.

An ***Information System*** can be defined as a subsystem of the business. Specifically, it is an arrangement of interdependent human and machine components that interact to support the operational, managerial, and decision-making information needs of an organisation.

Information systems consist of subsystems, including hardware, software, and data storage for files and databases. The set of subsystems that uses - the specific equipment, programs, files, and procedures - constitutes information systems application. Thus, information systems can be purchasing, accounting, payroll or sales applications.

1.2 Categories of Information Systems

Systems analysts develop different types of information systems to meet a variety of business needs. The types of Information systems and a brief description of each are listed below :-

1. Data Processing Systems

Processes large amount of data for routine business transactions, these systems run a series of programs on an automatic basis at regular intervals.
e.g. Payroll, inventory, financial accounting etc.

2. Management Information Systems [MIS]

Provides periodic reports for planning, control and decision making. Users of MIS use a shared database. MIS generates information that is used in decision making.

3. Decision Support Systems [DSS]

DSS supports decision makers by providing information on demand . DSS is similar to MIS, in that, they both depend on a database as a source of data. DSS differs from MIS in that it emphasises the support of decision making in all of its phases. Though, the actual decision is still in the hands of the decision maker.

4. Expert Systems

Expert systems capture expertise of decision makers in solving problems. They are a very special class of information systems made available for business, with the recent and widespread availability of hardware and software, such as, microcomputers and expert systems shells. An expert systems (also called **knowledge based system**) effectively captures and uses the knowledge of an expert for solving a particular problem experienced in an organisation. E.g. medical system, judicial system.

Unlike DSS which leaves the final judgement to the decision maker, an expert system selects the best solution available to a problem or a specific class of problems .

Data being processed in each of these systems may be done using any of the following modes:

- On-line processing
- Batch processing
- Transaction processing.

1.3 What is Systems Analysis and Design

Systems development has two major components, **Systems Analysis and Systems Design.** Systems analysis and design refers to the process of examining a business situation with the intent of improving it through better methods and procedures. Systems design is the process of planning a new business system to replace the old. But before this planning can be done, we must *thoroughly understand* the old system and determine how the computer can be best used to make its operation more effective.

Systems analysis then, is the process of totally understanding the current system by gathering and interpreting facts, diagnosing problems, and using the facts to improve the current system. This is the job of the Systems Analyst.

Having determined the requirements and ***'what'*** the system is intended to do, the systems designer *designs* the new system keeping in mind the objectives set during the systems analysis. This job is either done by the systems analyst himself or by another person called *systems designer*.

Thus, Analysis specifies **'what'** the system should do, that is it sets the objective and Design states **'how'** to accomplish this objective.

1.4 Systems Development Life Cycle (SDLC)

As already mentioned Systems development is a process consisting of the two major steps of Systems Analysis and Design. It starts when management or the systems development personnel realise that a particular business system needs improvement.

The Systems development life cycle is a sequence of events carried out by analysts, designers and users to develop and implement an information system. These activities are carried out in different stages.

Generally speaking, the phases of a project should be *completed* in sequence. Unfortunately most life cycles leave you with the impression that you must finish one phase or task *before* starting the next. In reality *phases of systems development life cycle canoverlap.* **Each phase is subject to change,** *based on the outcome of reviews held at the end of each phase.*

This section examines each of the seven events that make up the systems development life cycle. We will follow the waterfall life cycle approach. Most systems activities are all closely related and are usually inseparable. Even the order of the steps in these activities maybe difficult to determine. Different parts of the project can be in various phases at the same time. Some components maybe undergoing analysis while others are at advanced design stages.

Exercise - 1

1. What is a System ?
2. Analysis specifies ________ the system is and
   design specifies ________ to do it.

Ans: 1) A system is an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective.
2) What, how

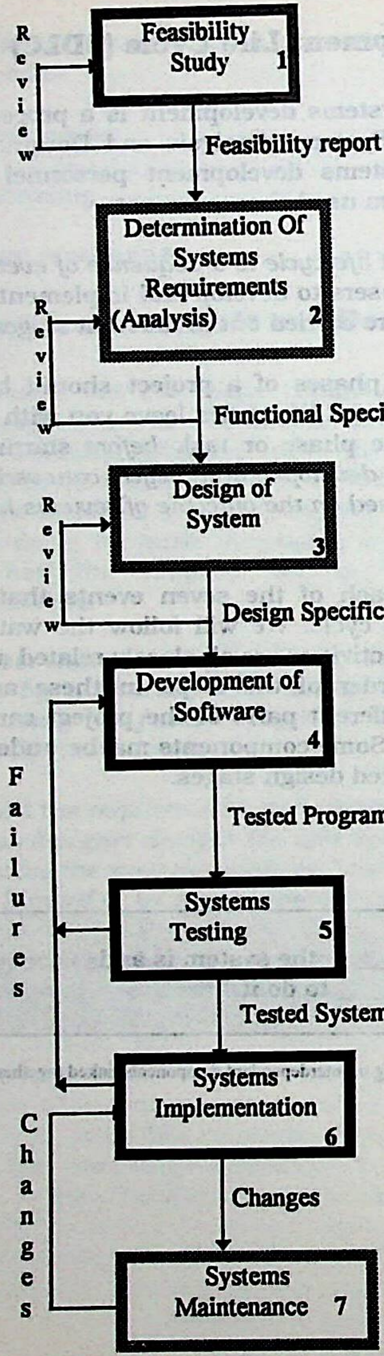


Figure 1.1 Phases Of Systems Development Life Cycle

The Phases are as follows :-

1. Preliminary investigation (feasibility study)

2. Determination of Systems requirements

3. Design of the system

4. Development of software

5. Systems testing

6. Systems implementation

7. Systems Maintenance

Here is an overview of each of these activities.

1.4.1 Preliminary Investigation (Feasibility Study)

One must know *what the problem* is *before* it can be solved. This phase starts as soon as someone, either a user or a member of a particular department recognises a problem or initiates a request, to modify the current computerised system, or to computerise the current manual system. When the request is made, the first systems activity, which is the preliminary investigation begins. **An important *outcome of the preliminary investigation is determining whether the system requested is feasible or not.***

The study is carried out by a small group of people who are familiar with information systems techniques, understand the part of the business or organisation that will be involved or affected by the project,. and are skilled in systems analysis and design process. Fact-Finding techniques are made use of to gather the required information. These are discussed latter in this session.

The major purposes of this study are listed below:

➢ **Identify the responsible users and develop an initial "scope" of the system.** This may involve conducting a series of interviews to see which user(s) are involved in (or affected by) the proposed project and which are not. The scope of the system is determined based on the information collected from the users. The scope determines the functionality of a system.

- **Identify current deficiencies in the user's environment.** This will usually consist of a simple narrative list of functions that are missing or operating unacceptably in the current system. The process of identifying deficiencies may also throw up the need for modifications/ enhancements to the existing system.

- **Determine objectives for the new system.** This may also be a narrative list consisting of existing functions that need to be re-implemented, new functions that need to be added, and performance criteria for the new system.

- **Determine whether it is feasible to automate the system and, if so, suggest some acceptable options.** This will involve some very crude and approximate estimates of the schedule and cost to build a new system.

The three major areas to consider while determining the feasibility of a project are:

Technical feasibility : The analyst must find out whether current technical resources which are available in the organisation is capable of handling the user's requirements.

If not, then the analyst with the help of vendors should confirm whether the technology is available and capable of meeting the user's request.

Economic feasibility : Economic or financial feasibility is the second part of resource determination. The basic resources to consider are :

- Management time

- Time spent by the systems analysis team

- Cost of doing the full systems study (including time of employees you will be working with)

- Estimated cost of hardware

- Estimated cost of software and / or software development

The concerned business must be able to see the value of the investment it is considering before committing to an entire systems study. If short term costs are not overshadowed by long term gains, or there is not an immediate reduction in operating costs, then the system is not economically feasible and the project should not proceed any further.

Operational feasibility : Once it is determined that the system is both technical and economically feasible then it has to be seen if it is operationally feasible. Operational feasibility is dependent upon determining human resources for the project. It refers to projecting whether the system will operate and be used once it is installed.

If the ultimate users are virtually weded to the present system and they see no problem and if they are *not* involved in requesting for a new system, then resistance to its operation will be strong. Chances for the system ever becoming operational are low.

Alternatively, if users themselves have expressed a need for an improved system, then they will put in all efforts to see that it becomes operational, and will eventually use it.

The document to be produced at the end of this activity is called **"Feasibility Study Report".**

Exercise-2

1. Write the full forms of
   DPS, MIS, DSS
2. Write the different stages of system development life cycle in sequence

Ans : 1) Data processing system, Management Information System & Decision Support System
2) Feasibility study, System requirement (analysis), System design, Development of software, Testing, Implementation, and Maintenance

We will now present a Case Study. The guidelines for preparing a feasibility report for this project is presented after the case study.

Case Study - Payroll System

For our case study, we have chosen the payroll system of ABC Co. Ltd., which is a very familiar and simple system. Reference to this case will be made through out the book.

☞ Organisational Overview Of ABC Co. Ltd.

"Payroll System" for ABC Co. Ltd.

ABC Co. Ltd. is the only sales outlet in Bombay for certain consumer goods. It basically consists of three departments.

1. **Sales department: Employees of this department are involved with the order processing system of the company. They carry out all the sales activities.**
2. **Accounts department : This department is involved with all the financial accounting aspects of the company.**
3. **Administration department: The major activity of this department, is payroll processing of all departments, and recruitment of new personnel.**

One of the jobs of administration department, is to calculate the payroll of the entire company. This so far is being done manually. The administration manager finds it very time consuming and feels that this system should be automated.

☞ Current manual system

At present the administration department maintains *three separate employee registers,* one for each of the three departments of the company. The *payroll is processed separately for each of the three departments.*

As and when any new employee joins, the *appointment form* containing all the employees standard details are sent to the administration department. These details are entered in the respective department's register.

A few months ago, a particular new employee - Anil's, details from the appointment form were entered by the administration clerk Joe, in the sales department register instead of the accounts department register.

This mistake was realised after the entire processing was complete. When the accounts department could not locate Anil's payslip, they approached the administration department. Joe, who had actually entered the register was absent. Another clerk Kumar, denied receiving Anil's appointment form. He checked the accounts department's employee register and argued that if it had come then the details would have been written into this register. The details were finally entered in the accounts department employee register, and the payslip for Anil was prepared.

After a week when Joe resumed duty he was informed by Kumar as to how he had to oblige the accounts department by preparing Anil's payslip after all the work was done. At that time Joe realised his mistake. He then had to cancel the information from the sales department employee register. Anil's payslip which had actually been prepared and sent to the sales department was found lying on the desk of the sales clerk who had distributed the payslips.

Just as the appointment form is sent to the administration any changes in the employee details are also sent by the respective departments, and these changes are incorporated in the respective registers.

By the 20th. of every month, the departments send in their attendance registers and overtime registers. The accounts department in addition, sends in the advance salary voucher details of *all* employees.

Using this information Joe and Kumar work out No. of days absent, total overtime hours for the month and total advance salary taken by each employee.

There have been many instances when these figures have been wrongly calculated, resulting in under or over payment. Naturally, the under payments have been reported while only few over paid cases have been reported.

Having completed this, the payroll of each department is calculated separately using the employee register details, days absent details, total overtime hours and advance salary details, the monthly payroll statement for each department is prepared. This statement is used as basis for preparing payslips, bank statement and salary summary statements.

Again there are a number of instances when:

1. The figures appearing in the payslips do not tally with the payroll statement or the bank statement.

2. The figures appearing in the bank statement are wrong.

3. The salary summary statement does not tally.

All this causes a great amount of chaos and confusion every month. All these problems were reported to the management on various occasions. It was finally decided to computerise the Payroll Monitoring function of the company.

'Success Consultants' were entrusted with the development of the entire Payroll system of the company. They were asked first to carry out a feasibility study and hand over a feasibility report to the management.

Guidelines For Preparing A Feasibility Report For The Automation Of The Payroll System For ABC Co. Ltd.

☞ **Identify the Responsible Users and Develop an Initial 'Scope' of the system**

The analyst must identify two specific groups of end-users:

a. Those who use the system. In this case the officers and clerks who actually collect the data and calculate the payroll.

b. Those affected by the inputs and outputs of the system in study. In this case the Accounts department who should receive the accounts statement, and all those in the administration department who are involved with the inputs and outputs.

To develop the initial scope of the system you need to get a broad idea of the system in study. In this case we can identify the main process as 'the payroll monitoring process', which gets its inputs from the three departments of the company, which are the sales department, accounts department and the administration department. The outputs of the systems are sent back to the respective departments.

Besides, we need to develop an initial context diagram -which is a simple data flow diagram in which the entire system is represented by a single process. (The context diagram is described in chapter 2).

☞ Identify current deficiencies in the user's environment

As the payroll processing is being done separately for each of the departments, there is duplication of registers, and processes.

As there are separate registers maintained, very often the entries are entered in the wrong registers, causing duplication of work and confusion.

Errors in calculation result in employees being wrongly paid, which need to be rectified in the following month.

Payslips and all the required reports have to be typed out. This is not only very laborious and time consuming but there are a number of errors found. Very often the statements do not tally.

☞ Determine Objectives for a new system

Here we will briefly list the functions of the new system.

- ➢ Maintenance of a employee details in a central location.
  Entry of new employee details.
  Updation of old employee details.

- ➢ Maintenance of a transaction details, and calculation of the days absent and Total overtime hours of the month using the attendance details and daywise overtime details obtained from the departments.

- ➢ Maintenance / check list printing o f current month's transaction details.

- ➢ Calculation of payroll and generation of the payslips.

- ➢ Generation of monthly reports :
  - Payslips
  - Cheques
  - Bank Statement (Payment advice)
  - Salary Summary Statement
  - Accounts Statement (for accounts)
  - PF Statement

- Generation of Yearly Reports:
  - Consolidated salary statement
  - Bonus Statement

- Adhoc reports (as and when required)
  - List of employee of particular grades
  - List of employees whose basic salary is between the given range.

Determine whether it is feasible to automate the system

The Analyst discusses goals and objectives for the new system in a review with the administration manager, officers, clerks handling the payroll and the company manager.

The administration manager considers the following three major areas to determine the feasibility of this project.

Technical feasibility : He determines whether the current level of technology can support the proposed system. ABC Co. ltd., already has hardware installed. This, at present is being used by the accounts department. They are in a position to spare one terminal to the administration department. The current set up is sufficient for the processing of the payroll once a month and even for adhoc reports as well as the annual reports.

Economic feasibility: He measures the cost effectiveness of the project. He now need not invest in the hardware as it is already available. He will still need to consider the time spent by the systems analysis' team, the cost of doing a full systems study, cost of employee's time involved in the study and the cost of the development of the software which has been entrusted to 'Success Consultants' .

Operational feasibility : He considers the extent the proposed system will fulfil his department's requirements. That is whether the proposed system covers all aspects of the working system and whether it has considerable improvements. In this case the employees of the administration department themselves have made the request to computerise the payroll system. They are very keen to see it operational.

On having decided that they should go ahead, they request 'Success Consultants' to carry out the Analysis and Design of the company's payroll system.

1.4.2. Determination of Requirements (Analysis)

After the feasibility study is done, a formal acceptance of the proposed system is taken from the user. The *feasibility report* is then taken as the basis for the next activity.

The next activity is **Determination of Systems requirements,** which involves studying the current business system *in great detail*, to find out how it works and where improvements hve to be made. A *requirement is a feature that must be included in a new system.*It may include a way of capturing or processing data, producing information, controlling a business activity, or supporting management.

The determination of requirement thus includes studying the existing system and collecting data (using the fact finding techniques discussed in section 1.5) about it to find out what the requirements are. This activity maybe carried out in two phases.

a. Detailed investigation

The heart of systems analysis is aimed at having a detailed understanding of all the important facets of the project under consideration. Analysts working closely with employees and managers must be able to answer the following key questions :-

i) What is being done by the current system?

ii) How it is being done ?

iii) How frequently does it occur ?

iv) How big is the volume of transactions or decisions ?

v) How well is the task being performed ?

vi) Does a problem exist ?

vii) If a problem exists, how serious it is ?

viii) If a problem exists, what is the underlying cause ?

To answer the above questions systems analysts talk to a variety of people to gather details about the project. Questionnaires are used to collect this information from large groups of people who cannot be interviewed individually.

Detailed investigations also require the study of manuals and reports, actual observation of work activities and collection of existing forms and documents to fully understand the project.

As the details are gathered, the analyst studies the requirements and identifies features the new system should have. The features include the information the system should produce and operational features such as processing controls, response times and input and output methods.

b. Analysis or Determination of systems requirements

After having understood the system, the next phase is to carry out a detailed study of the various operations performed by the system and their relationships within and outside the system. It is during this phase that the analyst and the user come to an agreement on what functions the proposed system has to perform.

A detailed document has to be prepared by the Systems Analyst containing the following :-

1. Inputs that must be received by the system.
2. The outputs to be produced by the system.
3. The data to be retained.
4. The procedures to get the output from the given inputs.
5. Audit and Control requirements - This would specify the features/ functions/ procedures that are required for the user to monitor and ensure that the new system is working properly or not.
6. System Acceptance Criteria - This would list the tests that the user would actually perform to check if the system is acceptable or not.

This detailed document is called the **Functional Specification** or **Proposed Procedures.**

At the end of this phase, the analyst should conduct a walkthrough with the users to review the specifications for various aspects of the analysis.

1.4.4. Design of the System

Once analysis is completed, the analyst has a firm understanding of *what is to be done.* The next step is how the problem can be solved. The design of a system uses the *Functional Specification* as basis and produces the details that state ***how*** a system will meet the requirements identified during systems analysis. The design process should take care of the following:-

- Identification of reports and outputs the new system should produce.
- Scrutinise the data present on each report/output.
- Sketch the form or display as expected to appear at the end of completion of the system. This maybe done on paper or on a computer display, using one of the automated system design tools available.
- Description of data to be input, calculated or stored.
- Individual data items and calculation procedures are written in detail.
- The procedures written should tell how to process the data and produce the output.

The document produced at the end of this activity is called the **"Design Specification".** This document should have charts, tables and special symbols to portray the design.

Designing tools are used to facilitate analysis and represent the system diagramatically. These are discussed later in the book.

A structured-walkthrough, a method for reviewing the specifications for various aspects of a design, is a commonly used technique for making sure that the design is appropriate.

The *detailed design specification is passed onto the programmers for software development to commence.* Designers are responsible for guiding the programmers through the design specifications.

1.4.5. Development of Software

The "*Design Specification*" contains *program specifications* as one of its topics. This is used by programmers for the development of software. In this stage, the actual coding/writing of the programs is done. In some firms, separate groups of programmers do the programming whereas other firms employ analyst-programmers who do analysis and design as well as code programs. Programmers are also responsible for documenting the program including comments that explain both how and why a certain procedure was coded in a specific way. Programs are individually tested using some test/dummy data. Documentation is also essential to test the program and carry out maintenance once the application has been installed.

This activity of the systems development lifecycle produces tested programs.

1.4.6. Systems Testing

Once the programs are tested individually, then the system as a whole needs to be tested. During testing, the system is used experimentally to ensure that the software does not fail i.e. that it will run according to its specifications and in the way the users expect it to. Special test data (which should include all types of data) is prepared as input for processing and the results are examined to locate unexpected results. In many organisations, testing is performed by persons other than those who wrote the original programs. Using persons who do not know how the programs were designed or programmed ensures more complete and reliable software. Various methods are available for testing. These are discussed later in the course.

This phase of the SDLC produces the tested system.

1.4.7. Systems Implementation

In this stage, the systems analysts put the *new software which has been tested* into use. User personnel are trained and any files of data needed by the new system are constructed. In short, the new software is installed and then used.

1.4.8. Systems Maintenance

Once installed, the software is often used for many years. However, both the organisation and the users change. The environment may also change over a period of time. Therefore the software has to be maintained i.e. modifications and changes will be made to the software, files or procedures to meet the users requirements.

1.5 Fact Finding Techniques

As already seen Data collection is an important part of analysis. This can be accomplished using fact-finding techniques to gather information from the users. The following fact-finding techniques are used for this purpose . Each of them have been discussed briefly :-

- Interviews
- Questionnaires
- Record inspections (on- site review)
- Observation

Normally more than one of these techniques are used to ensure accurate investigation. Important points of each of these techniques are listed below.

1. Interviews

Analysts use interviews for Collection of information from individuals or groups who are generally current users of the existing system or potential users of the proposed system. They maybe managers or employees who provide data for the proposed system or who will be affected by it.

Interviews is a time consuming method. It is the best source of qualitative information (opinions, policies, and subjective descriptions of activities and problems).

Interviews allow analysts to discover areas of misunderstanding, unrealistic expectations and even indications of resistance to the proposed system.
Interviews could be:

1. Structured
Structured uses standardised questions. These could be:-
a. Open response format
Here the questions are answered in ones own words.
b. Close response format

Here set of prescribed answers are used.

Or

2. Unstructured
Here the questions are worded to suit respondent may produce information about areas overlooked and which could be important.

2. Questionnaires

The use of questionnaires allows collection of data from large number of persons.

Standardised question formats yield more reliable data where the responses could be more honest. In this method the Analyst cannot observe reactions or expressions of respondents.

Questionnaires could be Open-ended OR Closed questionnaires. Open-ended questionnaires are used to learn feelings, opinions, general experiences on process detail or problem.

Closed questionnaires offers specific responses which have to be selected.

Questionnaires need to be printed and hence its a costly method.

3. Record Review

Many kinds of Records and reports provide valuable information about organisation and operations. Records include written policy manuals, regulations, standard operation procedures used by the organisation as a guide. These do not show actual activities, where decision making power lies or how tasks are performed. They Help in understanding the system. While observing the current reports one should scrutinise the data present in them.

4. Observation

This method helps to obtain first hand information on how activities are being carried out. It is useful when analysts needs to observe how processes are carried out, and whether specified steps are actually followed. One should know what to look for and how to assess the significance of what is observed.

1.6 Job Of A Systems Analyst

We have briefly discussed the various phases of the System Development Life Cycle. A number of people are involved in the process at different stages - the systems analyst, designers, programmers, users, operators. Systems analysts and design is carried out by a systems analyst.

A Systems Analyst studies the problems and needs of an organisation to determine how people, method, and computer technology can best accomplish *improvements* for the business. When computer technology is used, the analyst is responsible for the *efficient capture of data* from its business source, the *flow of that data to the computer*, and the *timely information* back to business users.

1.6.1 Qualities of Systems Analysts

Proficiency in systems analysis and design comes with experience and time. However, there are some essential qualities that a Systems Analysts should posses:

- The analyst is a ***problem solver***. He/she views the analysis of problems as a challenge and outputs a workable solution.
- The Analyst should be ***capable of tackling situations*** at hand through skilful application of tools, techniques and experience.
- The Analyst must be a ***good communicator*** capable of relating meaningfully to other people over a length of time, to gain information requirements from users and to communicate well with programmers.
- The Analyst needs enough ***computer experience*** to program and to understand the capabilities of computers.

Success of a system depends largely on how well it is understood and interpreted. **Thus, *communicating and dealing with people is a very important part of the Systems Analysts' job.***

1.7 Members Of The Systems Development Team

Traditionally Systems Analyst and Systems designers served as communication links between the programmers and the user/management group. **Analysts** helped users specify their information requirements. **Designers** turned those requirements into technical specifications and **programmers** turned the technical specifications into working programs for the user/management group.

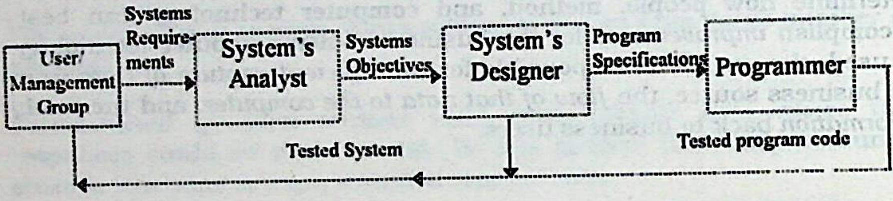


Figure 1.2 Shows The Link Between The Development Team

1.8 Introduction to Software Engineering

Software Engineering is a field of computer science that deals with the building of large complex systems. These systems are not built by a single person; they are built by a team or several teams of software engineers.

Software Engineering is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures and associated.

This definition of software engineering contains two key points. Firstly, software includes a good deal more than just computer programs. Thus, learning to be a good software engineer means a good deal more than learning how to generate computer programs. It also involves learning the skills to produce good documentation, data bases, and operational procedures for computer systems. The "mathematics and science" of software engineering is to provide methodologies for developing software as close to the scientific method as possible.

Secondly, the software products must be useful to man. The phrase "useful to man" emphases the needs of the user and the software's interface with the user. Software engineers must apply their skills and judgment to the job of developing appropriate set of specifications, and ensuring that the resulting software does indeed make the computer equipment perform functions useful to society. Also for something to be useful to people, it must satisfy a human need at a cost that society can afford.

The basic goal of software engineering is to produce high quality software at low cost. There are various techniques available to estimate cost and also measure the quality of a software product.

A software system is a component of a much larger system. The software engineering activity is a part of the design of a larger system. There are various approaches to the development of large systems. These are discussed in the next section.

1.9 Computer Aided Software Engineering (CASE)

Software development and maintenance is a mammoth task especially if done manually. CASE is the technology for automating software development and maintenance. The basic idea of CASE is to provide a set of well-integrated, labour saving tools and automating all phases of the software development life cycle. Hence, CASE is a tool for software engineering. CASE tools are software tools that support activities such as analysis, design, project management and maintenance. There are CASE tools available today that support most of the methodologies. Examples of CASE tools are EXCELERATOR, Information Engineering Workbench (IEW), Turbo Analyst.

1.10 Software Methodologies

The "waterfall life cycle" which was developed in the late 1960s is one of the commonly practiced approach to software development till the early 1990s. The waterfall life cycle is illustrated below

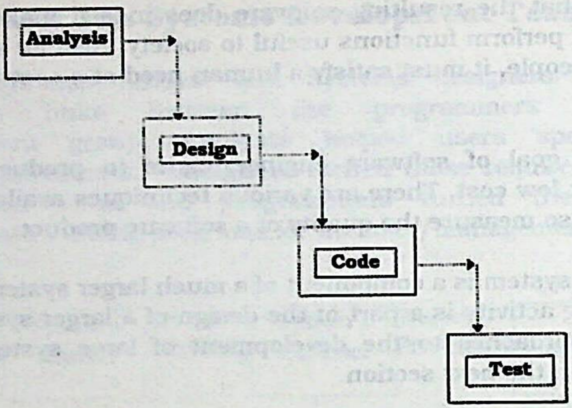


Figure 1.3 The Waterfall life cycle

Today we have number of other approaches and methodologies being proposed by various people.

Method vs. Methodology

With the above model as an example, we can discuss the difference between the terms *"method"*, *"methodology"*, and *"life cycle"*. Different organisations use these terms differently and there are various interpretations of these words.

Methodology - is a step-by-step plan for achieving some desired result. A software methodology usually identifies the major activities (for example, analysis, design, coding, and testing) to be carried out and indicates which people (users, managers, technicians) should be involved in each activity and what role they play. Methodologies often describe entry criteria, exit criteria and checkpoints for each of the activities/stages. The term **Life-Cycle** can be used synonymously with the term Methodology.

Method - is a step-by-step technical approach for performing one or more of the major activities identified in the an overall methodology. Thus, "structured analysis" is a method for carrying out the analysis phase of a project, while "object-oriented design" is a method for performing the design phase. (These different approaches are discussed later).

In many cases method and methodology go hand in hand and can be used in combination also. Like the waterfall life cycle is completely method independent in the sense that it can be used with an information engineering method, a structured analysis/design method or an object-oriented analysis/design method. Hence, it is a misconception to say that an organisation cannot follow the object-oriented approach because it is following the waterfall life cycle. As is commonly misunderstood, the waterfall approach is not opposed to any method.

Popular Methodologies

The most popular methodologies in use today are based on structures techniques or information engineering techniques. Object-oriented techniques have of late attracted a great deal of attention and may become the dominant methodology by the end of this decade.

Structured Methodologies

Structured techniques made their first appearance in the late 1960s, with the introduction of structured programming. Structured design and structured analysis techniques were proposed later in the 1970s. These techniques are best suited for process-oriented systems i.e. they are driven by the functionality (processes) of the system rather than by the data used in the system. It is more a function-oriented methodology than a data-oriented methodology. Structured analysis and design tools were originally characterised by two graphical modeling tools (data flow diagrams and structure charts) that emphasised the functions performed by a system. Additional components included data dictionaries, process specifications and later on incorporated entity-relationship diagrams and state-transition diagrams to help the software engineer model the data and time-dependent behaviour of a system. The methodology aspect of structured techniques was first described by DeMarco as a progression from a "current physical" model to "new logical" model.

The Yourdon, DeMarco and Gane-Sarson methodologies follow the structured analysis and structured design approach. Most of the CASE vendors based their automated tools on the methodology described in the books written by Yourdon & Constantine, DeMarco and Gane & Sarson.

Jackson's Data Design Methodology

In the Jackson approach, the input and output data structures are defined first which are then integrated to define the processes. The emphasis in this methodology is on data rather than process.

Warnier-Orr Design Methodology

This is a data oriented approach where the focus is on system output. The software engineer first defines the outputs required by the system and then works backwards and develops the system model. This system model will define all the inputs required and the processes required to transform them.

Information Engineering (IE) Methodologies

The IE methodologies reverse the emphasis : data plays the dominant role and functions play a subordinate role. This has been promoted by people like James Martin and CASE vendors like Texas Instruments, KnowledgeWare etc. This methodology includes business functions/entity-type matrices to show relationships within the enterprise and entities they use or modify and include diagrams like E/R diagrams, hierarchy diagrams etc.

More important than the emphasis on data is the emphasis on the level of the model in IE methodologies. While structured methodologies can be used to model entire enterprises, in the majority of cases they are used to model individual programs or systems. IE is perceived as a higher-level methodology that is intended to first model the enterprise/organisation and then model the individual systems that constitute the organisation. The idea is not to study only a particular system but the organisation as a whole which constitutes a number of subsystems. Information is the thread binding all the different functions/subsystems and hence it forms the basis for analysing the business needs, objectives and strategies. An organisational view is taken and a model built which shows the relationships of these various elements (Entity-Relationship model). This model when further decomposed will give details of the individual systems and the processes within them. Thus, IE is becoming gradually more popular as organisations are looking at enterprise wide solutions and total methodologies.

Object-Oriented Methodologies

Object-oriented programming technologies existed in 1960s (SIMULA language), but have gained more popularity recently and have been emphasised with object-oriented analysis and design techniques proposed by Grady Booch, Peter Coad and Edward Yourdon. The recent shift in interest to an object-oriented approach is a result of a gradual shift in priorities and technologies. Some of the key factors are :

The underlying concepts of an object-oriented approach have matured from issues of coding to issues of analysis and design.

The underlying technology for building systems has become much more powerful. Like it was difficult to think about coding in an object-oriented fashion when the language of choice was COBOL, FORTRAN, or C; it has become easier with C++, Objective-C, Smalltalk and Ada.

The systems we build today are different than they were 10 or 20 years ago. In every respect, they are larger and more complex; they are also more volatile and subject to constant change. We find that today's on-line, interactive systems devote much more attention to the user interface than the text-oriented batch processing systems. Almost 75% of the code in modern systems is concerned with user interfaces and this is particularly evident with the GUIs available today. With systems being subject to constant change, reusing components of existing systems and building newer systems has become a necessity. An object-oriented approach to such systems (from analysis through design and into coding) is a more natural way of dealing with such user-oriented systems. Systems built today are more "data-oriented" than systems built earlier. Hence, modeling the data has become a higher priority.

Which is the most suitable methodology?

Software engineers are constantly faced with the dilemma of which methodology to use for which system. The solution is to use the methodology depending on the application/system in question. The type and requirements of the application will dictate which method is best suited for that system. For example, the Yourdon method of structured techniques is very effectively used for process-oriented commercial applications like payroll, accounting etc.

Exercise - 3

1. The three major areas to consider while determining the feasibility of a project are _________, _______ & _________.
2. List four fact-finding techniques.
3. During analysis stage, a detailed document prepared by the system analyst is called _________________.
4. List the members of the system development team.
5. What is the most important partof a systems analysts' job.

Ans: 1) Operational, Technical and Economical
2) Interview, Questiorinaires, Record review and Observation

3) Functional Specification

4) User/ management, Systems analyst, Systems designer and Programmer.

5) communicating and dealing with people is a very important part of the Systems Analysts' job.

Summary

In this chapter we have learnt the following:-

✈ A **system** is simply a set of components that interacts to accomplish some purpose.

✈ An **information system** is the means by which data flows from one person of department to another.

✈ **System's analysis and design** is a series of processes systematically undertaken to improve a business through the use of computerised information systems. A large part of system's analysis and design involves working with current and eventual users of information systems.

Analysis specifies **'what'** the system should do.
Design states **'how'** to accomplish the objective.

- The various categories of information systems are :
  ✈ Data processing systems
  ✈ Management information systems (MIS)
  ✈ Decision Support Systems
  ✈ Expert Systems

- Systems development life cycle is a phased approach to analysis and design that holds that systems are best developed through use of a specific cycle of analyst and user activities.

The systems development life cycle method consists of the following activities:

1. Preliminary investigation (feasibility study)
2. Determination of Systems requirements
3. Design of the system
4. Development of software
5. Systems testing
6. Systems implementation
7. Systems Maintenance

Data collection being an important part of analysis, should be accomplished by fact-finding techniques. They are as follows :

- Interviews: structured or unstructured
- Questionnaires
- Record inspections
- Observation

Remember that time is money

- Benjamin Franklin

Chapter 2

Structured Analysis

At the end of this session, you will be able to :

- *Define structured analysis and its components*
- *Define data flow analysis and understand the tools used for data flow strategy*
- *Draw DFDs - context diagrams and levelling of DFDs*
- *Define Data Dictionary, know its importance and what it records*
- *Know what is prototyping*

2.0 Introduction

This chapter focuses on Structured Analysis. Apart from being introduced to "what" is Structured analysis, you will also learn about its components. You will be given an idea of Data Flow Analysis, Data Flow Strategy and Data Dictionary's. These various tools used for structured Analysis are explained in detail. You will be taught how to draw a Data Flow Diagram, the various symbols used in it, and also what is a data dictionary, its importance and what it records. Besides all this, it introduces to Prototyping.

2.1 What is Structured Analysis ?

Structured Analysis is a development method for the Analysis of existing, manual or automated systems, leading to the development of specifications (i.e. Design) for a new or modified system. Structured Analysis allows the analyst to learn about a system or a process in a manageable and logical way.

The objectives in Structured Analysis is to ***completely understand the current system, from which requirements are determined,*** which become the basis for a new or modified system. Fully understanding large, complex systems maybe difficult, but, structured analysis development method is aimed at overcoming this difficulty through its components which are described later in this chapter.

Structured Analysis came about during the late 1970s as a method of communicating more effectively with users during the entire system development life cycle. If structured analysis is done in a proper manner, it allows the systems analysts to develop systems that are wanted by the users and that can be used by them in an effective manner. The aim of structured analysis is to clearly define **WHAT** a systems requirements are.

2.2 Components of Structured Analysis

Structured Analysis uses the following components :-

1. Graphic Symbols

These include icons and Data flow diagrams. Instead of words, structured analysis uses symbols or icons to create a graphic model of the system. Graphic models show details of the system.

- **Icons** are pictorial representations of entities described by the data. Properly selected icons communicate information immediately. Icons eliminate the necessity for users to learn abbreviations, notations, or special nomenclature.

- **Data Flow Diagrams** is very important graphic tool which is used to describe and analyse the movement of data through a system - manual or automated. The model of the system is termed as data flow diagram.

2. Data Dictionary

The Data Dictionary stores details and descriptions of all data used in a system. It is an organised listing of all the data elements that are pertinent to the system.

3. Procedures and Process Description

This is the third major tool of structured analysis. The purpose of process description is to allow the analyst to describe the business policy represented by each of the bottom level bubbles in the bottom level DFDs. These can be written in a variety of forms such as - structured English, Decision trees, Decision tables, action diagrams. These are discussed later.

4. Rules

These are standards for describing and documenting the system correctly and completely. Good documentation will provide an explanation of how a system operates.

2.3 Data Flow Analysis

2.3.1 What is Data Flow Analysis

Systems analysis is basically centred around

→ What processes make up a system

→ What data are used in each process of the system

→ What data is stored

→ What data enter and leave the system

The emphasis is clearly on Data Analysis.

Following the flow of data through business processes is Data Flow Analysis. While handling transactions and completing tasks, data are input, processed, stored, retrieved, used, changed and output. Data flow analysis studies the use of data in each activity.

These findings are documented in

Data flow diagrams, which graphically show the relation between processes and data.

Data dictionary, which formally describe the systems data and where they are used.

Data flow analysis maybe thought of as viewing the activities of a system from the viewpoint of the data :-

- **Where they originate ?**
- **How they are used or changed ?**
- **Where they go including the stops along the way from their origin to their destination.**

2.3.2 Tools of Data Flow Strategy

The components of Data flow strategy are used for both systems requirement (analysis) as well as systems design. Data flow strategy makes use of the following tools :-

1. Data Flow Diagram

Through the use of a structured analysis technique called *data flow diagram,* the system analyst is able to put together a *graphical representation of data processes through the organisation.* The data flow approach emphasises the logic underlying the system. By using combinations of only four symbols, the analyst is able to create a pictorial depiction of processes that eventually can provide solid system documentation.

2. Data Dictionary

All definitions of elements in the system are described in detail in a data dictionary. The data dictionary is a reference work of data about data i.e. metadata, complied by the analyst to guide them through analysis and design.

3. Data Structure Diagram

Although historically, systems analysts have thought of data in the form of data items, records, and files, experience has shown that a broader framework is needed. Data structure diagrams, is a useful tool for developing this framework.

Data Structure Diagrams are graphic tools that show the logical data structure requirements of an information systems application. Each data item either identifies the entities or describes an important attribute. Data structure diagrams organise the data.

4. Structure Chart

This is a design tool that visually displays the relationships between program modules. It shows which modules within the system interact and also graphically depicts the data that are transmitted between various modules. Structure charts are developed before the start of coding programs. They don't express procedural logic nor do they describe the actual physical interface between processing functions. They identify the data passes existing between individual modules that interact with one another.

Data structure diagrams and structures charts are dealt with in subsequent chapters.

2.4 Data Flow Diagrams

Data flow diagram is a graphic tool. It is used to describe and analyse the movement of data through a system - manual or automated. They focus on the data flowing *into the system, between processes and in and out of data stores.* This is a central tool and the basis from which other components are developed. *The system models are termed as Data Flow diagrams (DFD).*

Developing a description of the system using structured analysis follows a *top-down process.* A full description of a system actually consists of a set of DFDs, which comprises of various levels. An initial overview model is *exploded* into more detailed lower level diagrams that show additional features of the system. Further each process can be broken down into a more detailed DFD. This occurs repeatedly until sufficient detail (lowest level) is described to allow the analyst to fully understand that portion of the system. The various levels of DFDs are shown later on in this section.

2.4.1 Types Of Data Flow Diagrams

DFDs are of two types.

1. Physical DFDs

Structured analysis states that the current system should be first understood clearly. The **physical DFD is a model of the current system** and is used to ensure that the current system has been clearly understood. Physical DFDs show actual devices, departments, people etc. involved in the current system.

2. Logical DFDs

Logical DFDs are the models of the proposed system. They should clearly show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the systems structure charts.

Both Physical and Logical DFDs support a top-down approach to systems analysis. For this purpose, Analysts begin by developing a general understanding of the system and gradually explode components in greater detail. This is achieved through the *Context diagram, First Level DFD, Second level DFD.* These concepts are explained in the following sections.

2.4.2 Drawing Data Flow Diagrams

2.4.2.1 Notation

DFDs are quite easy to read and understand. There are two alternative but equivalent symbol sets :

1. Yourdon symbol set
2. Gane - Sarson symbol set.

It is suggested that you do not mix and match symbol sets.

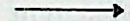
Four simple notations are sufficient to complete a DFD. They are :

1. Data Flow
2. Process
3. External entities
4. Data Store

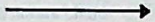
A brief description of each of these notations are given.

1. Data Flow

Data in a system move in a specific direction that is from origin to destination. The data flow is a *'packet'* of data indicating the movement of data within the system. Data flows must be inputs to or outputs from processes. They must contain data and all data flows should be labelled indicating what data is flowing. If the data flow is showing an input then the arrow should point towards the process, but incase the data flow is showing an output then the arrow should point away from the process.



Yourdon

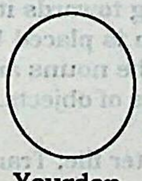


Gane - Sarson

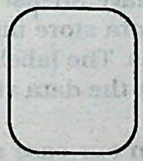
e.g. of Data flow : Month's Transactions details, New Employee Details.

2. Process

The emphasis in any DFD is placed on processing. Processes transform inputs into outputs. They are work or actions that are performed by people, machines, or computers on incoming data flows to produce outgoing data flows. They can be performed by people, departments, robots, machines, or computers. The details of the processing that is the ***logic or procedures are not shown in a DFD.*** The data flow leaving a process is *always* labelled differently from the one entering it. Each process should be given a meaningful name.



Yourdon



Gane - Sarson

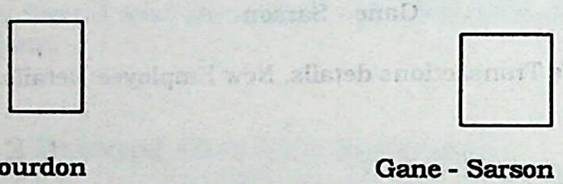
e.g. of Process: Calculate Net Pay, Preparation of payslips etc.

3. External Entities

External entities are organisations, other information systems, departments or people which represent a source or destination of transactions or data. When the system we are considering accepts data from another system or provides data to it, that other system is the external entity. By designating something or some system as an external entity, we are implying that it is outside the boundary of the system we are considering.

External entity represents any entity that supplies or receives information from the system but is not a part of the system. Each entity is labelled with a appropriate name. Although it interacts with the system, it is considered as external to the boundaries of the system.

Examples of entities in our case study are : New employee - from whom we obtain the employee's data; Departments - from where attendance and OT data are obtained; Accounts department - provides us with the advance salary data and this entity gets the accounts statement from the system.

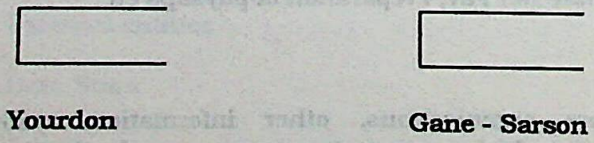


4. Data Store

Data stores could be thought of as the *'memory'* of the system. *Any place that data accumulates is a data store.* Data flow diagrams,*do not* specify the type of physical storage i.e. tape, disk etc. The data in a data store, are stored or referenced by a process in the system. They represent computerised or non-computerised devices.

A data store must have atleast one data flow pointing towards it, or one away from it. The data store must have a label which is placed between the two parallel lines. The labels of data stores must be nouns and must clearly identify what the data store contains as a class of objects.

E.g. of data stores in our case study : Employee master file; Transaction file; Payslip file, employee master register.



Exercise - 1

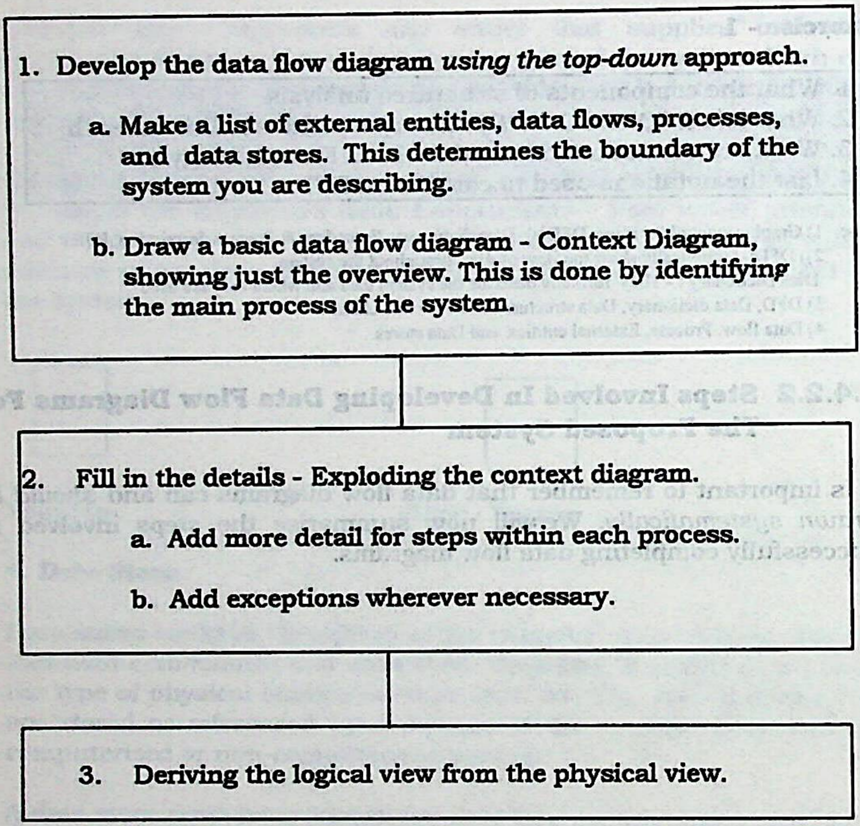
1. What the components of structured analysis
2. What is a DFD and Data Dictionary ? (write one line on each)
3. What are the main tools used for Data Flow Strategy
4. List the notations used to complete a DFD

Ans: 1) Graphic symbols -(icons, DFD's), Data dictionary, Procedure & Process description, Rules
2)) DFD:- Graphically show the flow of data throughout the system.
Data Dictionary : - They formally describe the system data and where they are used
3) DFD, Data dictionary, Data structures and Structure charts
4) Data flow, Process, External entities, and Data stores

2.4.2.2 Steps Involved In Developing Data Flow Diagrams For The Proposed System

It is important to remember that data flow diagrams can and *should be drawn systematically*. We will now summarise the steps involved in successfully completing data flow diagrams.

DEVELOPING DATA FLOW DIAGRAMS



1. Develop the data flow diagram *using the top-down* approach.

   a. Make a list of external entities, data flows, processes, and data stores. This determines the boundary of the system you are describing.

   b. Draw a basic data flow diagram - Context Diagram, showing just the overview. This is done by identifying the main process of the system.

2. Fill in the details - Exploding the context diagram.

   a. Add more detail for steps within each process.

   b. Add exceptions wherever necessary.

3. Deriving the logical view from the physical view.

Step 1. Using Top-Down Approach

You can begin a data flow diagram by first making a list, consisting of four categories :

- *external entities*
- *data flows*
- *process*
- *data stores*

This list helps determine the boundaries of the system you will be describing. Once a basic list of data elements has been complied, you can begin by drawing a context diagram. This is done by identifying the main process of the system. Each diagram should be self-contained on a single sheet of paper.

The initial context diagram should be an *overview* including basic inputs, processes, and outputs. This will be the most general diagram, really a *bird's eye view of data movement in the system. This is called taking a "top-down approach"* to diagramming data movement. With a top-down approach, the diagrams move from *general to specific.* While the first diagram helps the systems analyst grasp basic data movement, its general nature limits its usefulness.

Physical Context Diagram For Payroll Monitoring System

The DFD showing the general i.e. top layer of the system is called the 'Context diagram'.

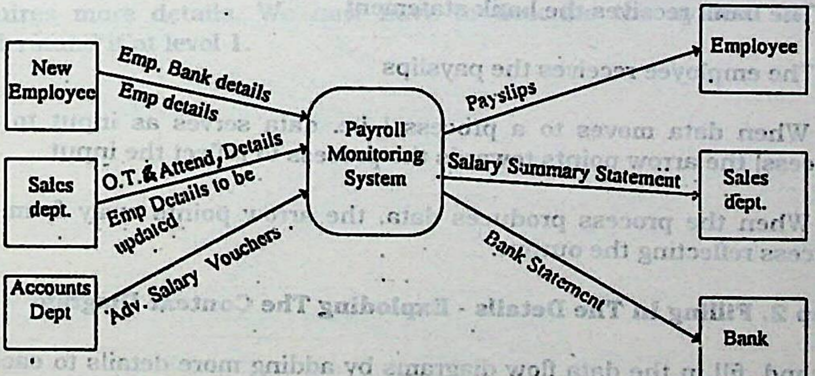


Fig 2.1

Figure 2.1 shows the Physical Context Diagram which describes the payroll monitoring system at a very general (top) level.

- The context diagram consists of only one process, data flows and entities. The main process of the system needs to be identified.

- The single process in figure 2.1 is 'Payroll Monitoring System'.

- The context diagram defines the system in study i.e. it determines the boundaries.

- Each arrow, representing data flow, is labelled to show what data are being used.

- New employee gives the new employee details and the employee's bank details to the system.

- The respective departments feed the system with the O.T details, attendance details and employee details which are to be updated.

- The accounts department give the advance salary details.

- The accounts department receives the accounts statement from the system.

- The salary statement is received by the respective department.

- The bank receives the bank statement

- The employee receives the payslips

- When data moves to a process (i.e. data serves as input to the process) the arrow points towards the process to reflect the input.

- When the process produces data, the arrow points away from the process reflecting the output.

Step 2. Filling In The Details - Exploding The Context Diagram

Second, fill in the data flow diagrams by adding more details to each of the processes. More detail is achievable through using a process called **"exploding the diagrams"**. (*Going from higher level to lower is called 'exploding' a data flow diagram.).*When the first diagram is made, inputs and outputs are specified and these remain constant throughout all of the following diagrams.

The original diagram is exploded into close-ups of three to nine processes (in our example we have exploded into five processes). New data stores and new data flows are added in lower levels. The effect is that of using a magnifying glass to the original data flow diagram. Each diagram should ideally use only a single sheet of paper. *By exploding DFDs into subprocesses, the systems analyst begins to fill in the details about data movement.*

Because the nature of complexity of systems vary, one cannot fix a specific number of levels of the DFD. Incase the DFDs are over complexed they cause difficulty in understanding them. On the other hand if they are under exploded, errors of omission could occur. You should go as far as necessary to understand the details of the system and the way it functions.

Care should be taken to verify all aspects with knowledgeable users by conducting a walk through with them. Changes need to be incorporated after getting users input.

Developing the First Level Physical Data Flow Diagram For Our Payroll Monitoring System

The description of the payroll monitoring in the context diagram requires more details. We now have to describe the system as we understand it at level 1.

FIRST LEVEL PHYSICAL DFD FOR PAYROLL CALCULATION OF SALES DEPT

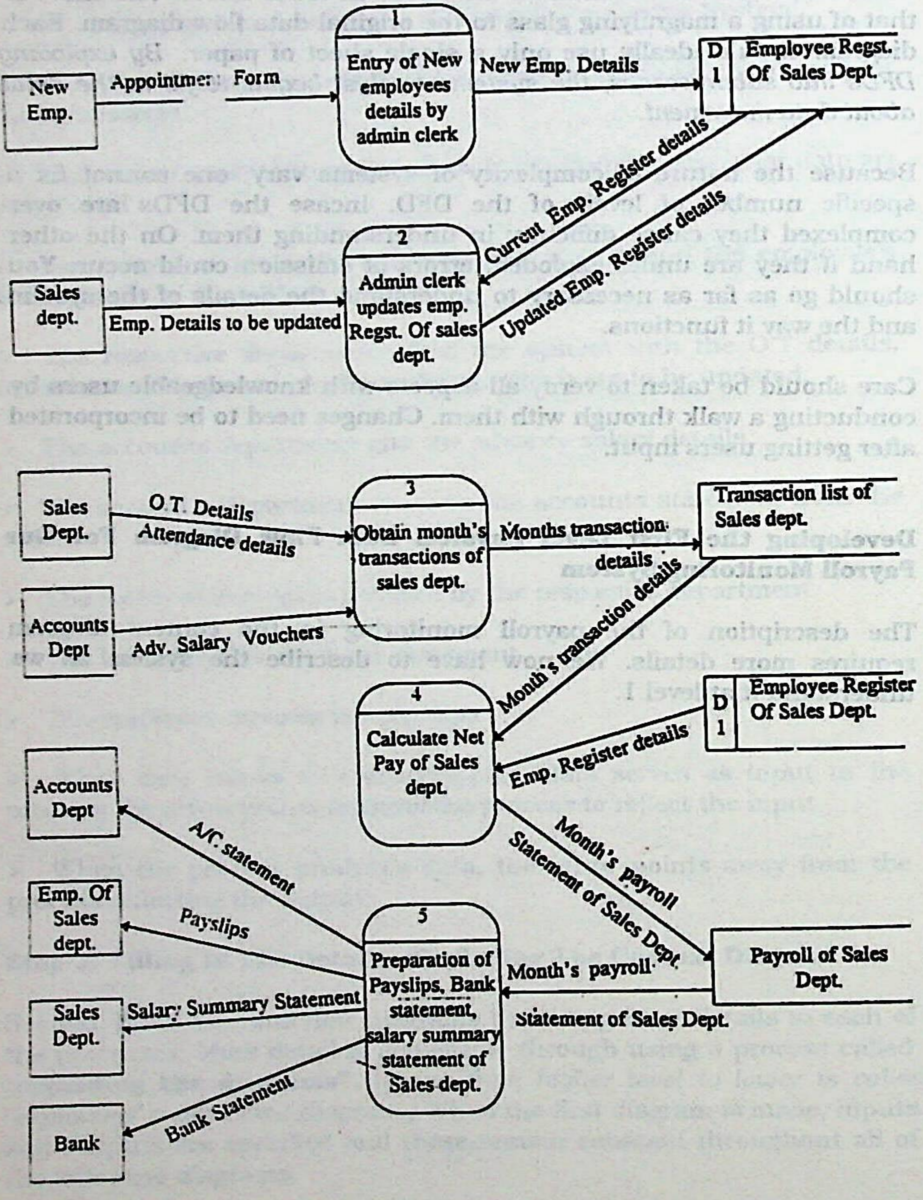

New Emp.
Appointmen: Form
1
Entry of New employees details by admin clerk
New Emp. Details
D 1
Employee Regst. Of Sales Dept.
Current Emp. Register details
Updated Emp. Register details
2
Admin clerk updates emp. Regst. Of sales dept.
Sales dept.
Emp. Details to be updated
Sales Dept.
O.T. Details
Attendance details
3
Obtain month's transactions of sales dept.
Months transaction details
Transaction list of Sales dept.
Accounts Dept
Adv. Salary Vouchers
Month's transaction details
4
Calculate Net Pay of Sales dept.
D 1
Employee Register Of Sales Dept.
Emp. Register details
Month's payroll Statement of Sales Dept.
Payroll of Sales Dept
Accounts Dept
A/C statement
Emp. Of Sales dept.
Payslips
5
Preparation of Payslips, Bank statement, salary summary statement of Sales dept.
Month's payroll Statement of Sales Dept.
Sales Dept.
Salary Summary Statement
Bank
Bank Statement

Fig. 2.2

Figure 2.2 shows the First Level Physical Data Flow Diagram of the payroll calculation of the Sales department of ABC Co. Ltd.

The same is applicable for the other two departments of the company. (only the department name will vary). As can be seen this level of the DFD contains five processes.

The following is seen in the DFD.

- The New Employee Data from the filled appointment form is entered into the respective department's employee register by the administration clerk.
- Administration clerk makes changes in the employee register wherever required.
- The month's transaction list is written out after the following gather the following details.

  * Month's attendance details are obtained from the attendance register.

  * Day-wise O.T data obtained from the O.T register

  * Advance salary details obtained from the advance salary vouchers obtained form the accounts department.
- Now using this transaction list and the details from the employee register the payroll of each employee is calculated. The payroll statement of the month is prepared using the calculated figures.
- The payroll statement forms the basis for typing out the payslips and the bank statement and salary statement.
- The payslips are handed over to the employees of the respective departments and the bank statement is sent to the banks.

Second Level Data Flow Diagram

We have just seen and discussed the First Level Physical DFD. We need more detail to understand the system better. The processes "obtaining month's transaction" and "Calculation" need more explanation. From that we learn that the processes need to be *exploded* in the sense, need for more detail is required to get a clearer idea of the system. We need to draw lower levels of the DFD to obtain this.

SECOND LEVEL DFD FOR OBTAINING MONTH'S TRANSACTIONS OF SALES DEPT.

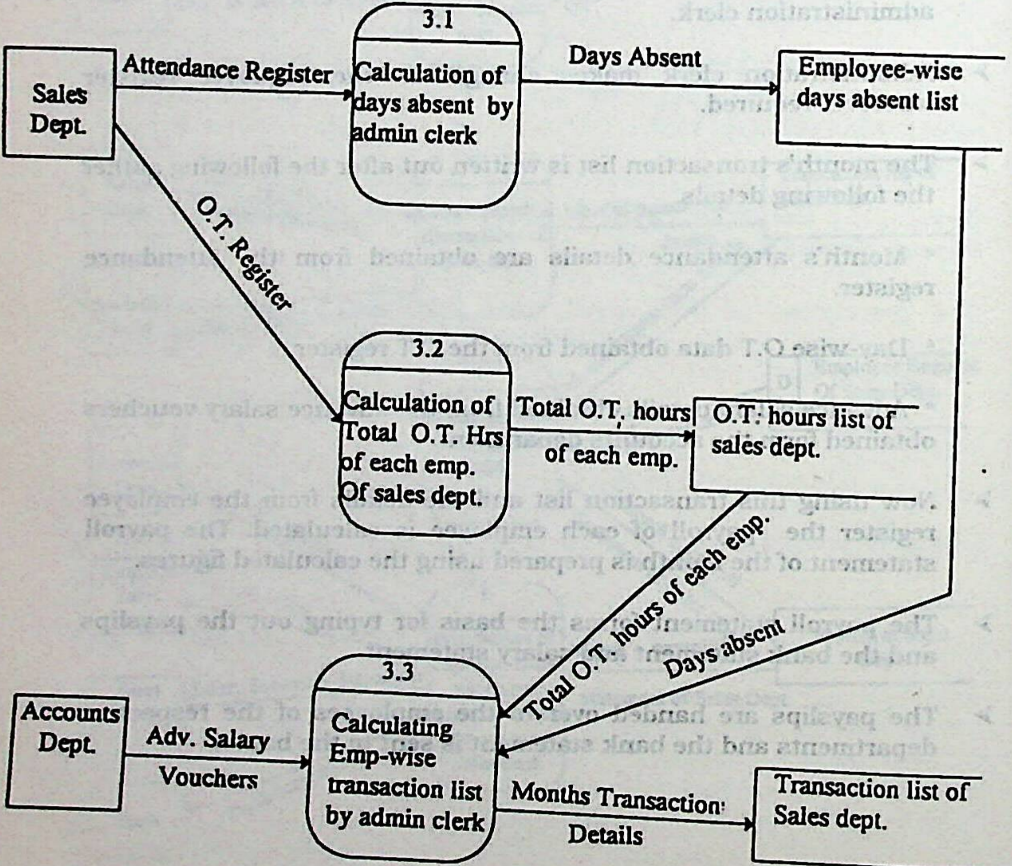

3.1
Sales Dept.
Attendance Register
Calculation of days absent by admin clerk
Days Absent
Employee-wise days absent list
O.T. Register
3.2
Calculation of Total O.T. Hrs of each emp. Of sales dept.
Total O.T. hours of each emp.
O.T. hours list of sales dept.
Total O.T. hours of each emp.
Days absent
3.3
Accounts Dept.
Adv. Salary Vouchers
Calculating Emp-wise transaction list by admin clerk
Months Transaction Details
Transaction list of Sales dept.

Fig. 2.3

Fig 2.3 shows the Second Level DFD for 'obtaining month's transaction' process for payroll calculation. From this we can gather :-

- That the administration clerk gets the attendance register from the respective department and calculates the days absent of each employee.

- He also gets the day-wise O.T register and totals the O.T hours of each employee.

- The accounts department supplies the admin. clerk with the advance salary taken details through vouchers.

- Having got this he makes out a month's transaction list containing days absent (if any), total O.T hours and advance salary taken for each employee.

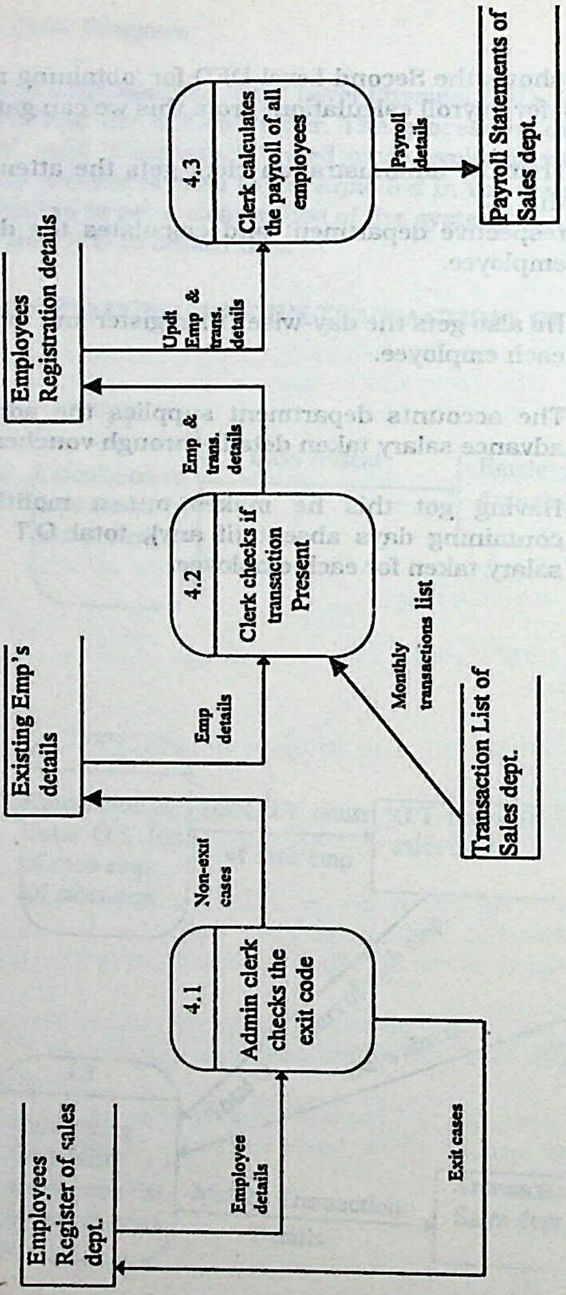

Employees Register of sales dept.
Employee details
Exit cases
4.1
Admin clerk checks the exit code
Non-exit cases
Existing Emp's details
Emp details
4.2
Clerk checks if transaction Present
Monthly transactions list
Transaction List of Sales dept.
Emp & trans. details
Employees Registration details
Updt Emp & trans. details
4.3
Clerk calculates the payroll of all employees
Payroll details
Payroll Statements of Sales dept.

Fig. 2.4 Second Level DFD for 'Calculation' process of the payroll system

From figure 2.4 we gather:-

- That the administration clerk checks the employee register and by-passes all exit cases.

- For each non-exit case he checks the transaction list for any transactions of that particular employee.

- He then uses the data present in the employee register and the transaction list (if data for the employee present) to calculate the payroll for the month. The month's payable data is thus obtained and stored in the payroll statement.

All activities, data flows and data stores in this lower-level view of the system must be included within the previous DFD.

In general, we should be certain of the following points:-

- All data flows that appeared on the previous diagram explaining the process are included in the lower level diagram.

- New data flows and data stores are added if they are used internally in the process to link processes introduced for the first time in the explosion at this level.

- Data flows and data stores that originate within the process must be shown.

- No entries should contradict the description of the higher-level data flow diagram (if they do, one or the other is either is incorrect or incomplete and a change must be introduced).

Notice how we continue using physical details of the process:

The clerk calculating the days absent using the attendance register, using the overtime register, typing of the transaction list, the clerk having to check the exit code and typing of the payroll statement and the payslips.

implementation (the users are better able to discuss the physical system as they know it thoroughly.)

A logical DFD also follows a top-down process. This involves drawing the context diagram and exploding it to obtain the first level, second level and so on until all the processes mentioned are clearly understood.

A logical DFD is derived from the physical version by doing the following :

- Showing the actual *'data'* needed in a process, not documents that contain them.
  e.g. show 'new emp. data' rather than 'appointment form'.
- Remove routing information, that is, show the flow between procedures' not between people, offices or locations.
- Remove tools and devices such as folders files etc.
- Consolidate redundant data stores.
- Remove unnecessary processes, such as those that do not change the data or data flows.

Figure 2.5 shows the logical Context Diagram For Payroll Calculation of the full company.

LOGICAL CONTEXT DIAGRAM FOR PAYROLL PROCESSING

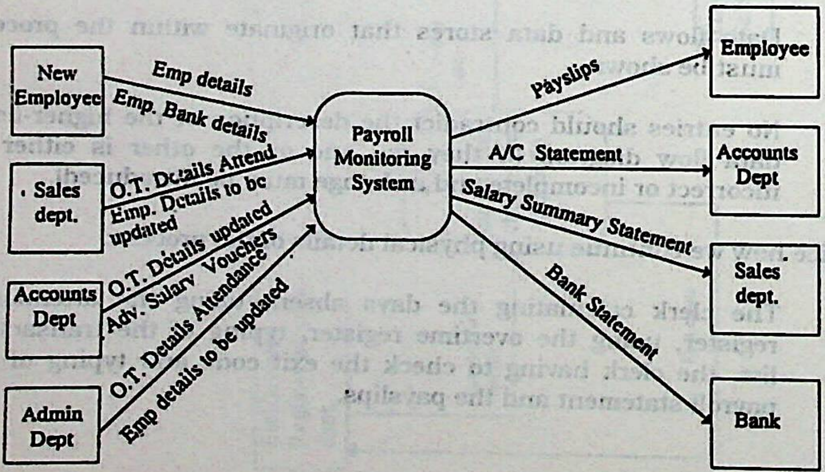


Fig 2.5

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

FIRST LEVEL LOGICAL DFD OF PAYROLL CALCULATION FOR ABC CO. LTD.

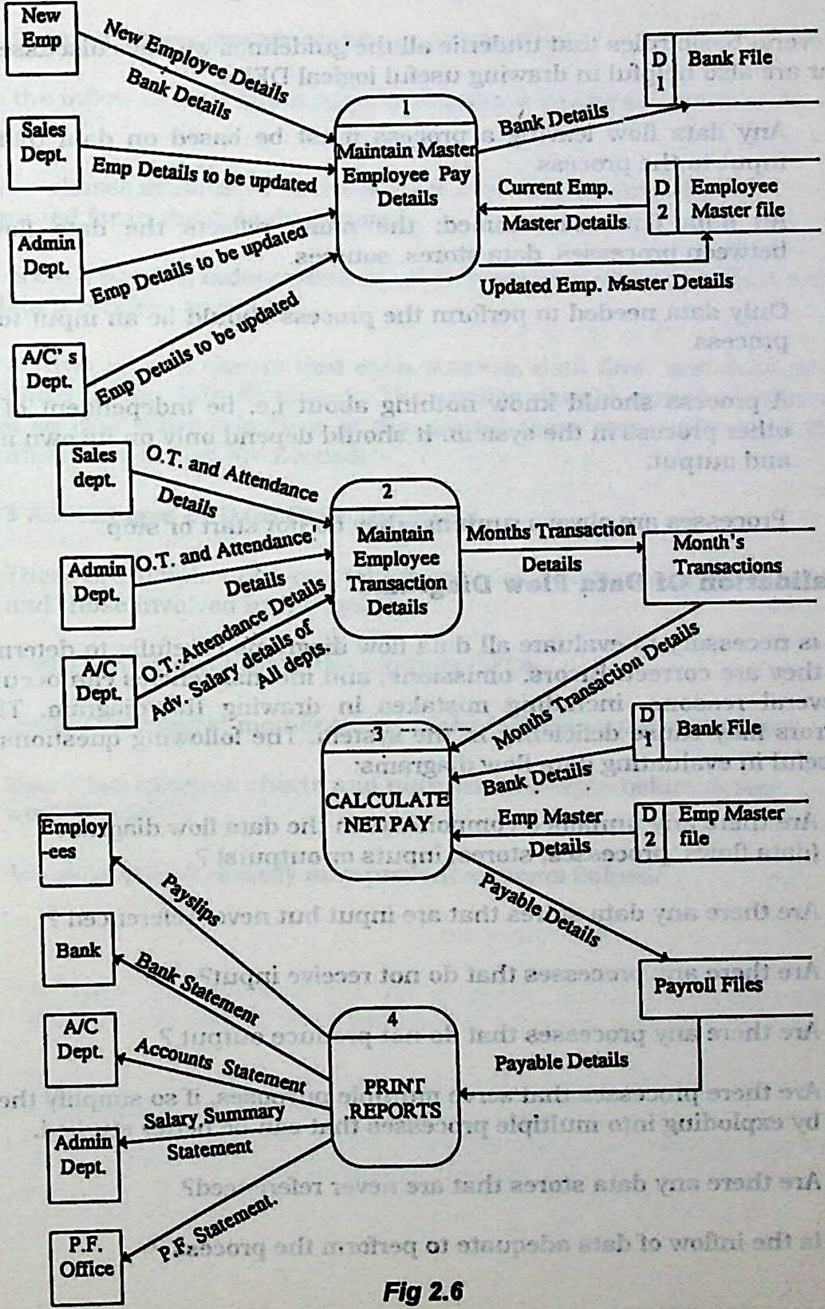


Fig 2.6

General Rules For Drawing Data Flow Diagrams

Several basic rules that underlie all the guidelines we have discussed so far are also helpful in drawing useful logical DFDs.

- Any data flow leaving a process must be based on data that are input to the process.
- All data flows are named; the name reflects the data flowing between processes, data stores, sources.
- Only data needed to perform the process should be an input to the process.
- A process should know nothing about i.e. be independent of any other process in the system. It should depend only on its own input and output.
- Processes are always running, they do not start or stop.

Validation Of Data Flow Diagrams

It is necessary to evaluate all data flow diagrams carefully to determine if they are correct. Errors, omissions, and inconsistencies can occur for several reasons, including mistakes in drawing the diagram. These errors may cause deficiency in the system. The following questions are useful in evaluating data flow diagrams:

1. Are there any unnamed components in the data flow diagram (data flows, processes, stores, inputs or outputs) ?
2. Are there any data stores that are input but never referenced ?
3. Are there any processes that do not receive input?
4. Are there any processes that do not produce output ?
5. Are there processes that serve multiple purposes, if so simplify them by exploding into multiple processes that can be better studied.
6. Are there any data stores that are never referenced?
7. Is the inflow of data adequate to perform the process ?

8. Is there excessive storage of data in a data store?

9. Is the inflow of data into a process too much for the øutput that is produced ?

10. Are aliases introduced in the system description? Are they accounted for in the data dictionary?

11. Is each process independent of other processes and dependent only on data it receives as input?

The analyst should assure that each process, data flow, and data store is defined in the data dictionary. The entries should contain sufficient detail so that other members of the project team can understand the definitions when they are needed.

2.4.3 Advantages of Data Flow Diagrams

- These are simple notations which are easily understood by users and those involved in the system.
- Users can be involved in the study of DFDs.
- Users can suggest modifications of the DFD for more accuracy .
- Users can examine charts and pinpoint problems before design work starts.
- Avoiding mistakes early may prevent systems failure.

2.5 Data Dictionary

The analyst should *assure that each process, data flow and data store appearing in the logical DFD is defined in the data dictionary.* The entries should contain sufficient detail so that other members of the project team can understand the definitions when they are needed.

We will now discuss what a Data Dictionary is?, Why is it important?, and What it records.

2.5.1 What Is A Data Dictionary ?

A Data Dictionary is a *catalogue* - of all elements in a system. It is a document that collects, co-ordinates, and confirms what a specific data terms mean to different people in the organisation. It is the basic reference work for finding the names and attributes of data elements used through out the system. These elements centre around *data* and the way they are *structured* to meet user requirements and organisation needs. It is a must that all data elements are included in the data dictionary. The major elements are *data flows, data stores, and processes.* Data Dictionary stores details and descriptions of these elements.

A well developed Data Dictionary should be able to provide the following information:

- How many characters are in a data item (data element, field) ?
- By what names it is referenced in the system?
- Where it is used in the system ?

Data flow diagrams, covered in the previous section are an excellent starting point for collecting data dictionary entries.

It is important to update the data dictionary as changes occur. The dictionary is developed during the Data Flow Analysis and assists the analysts involved in determining systems requirements, however its contents are used during systems design as well.

2.5.2 Why Is A Data Dictionary Important ?

The data dictionary is important for the following reasons:-

1. To Manage the details

Both large and small systems have large quantities of data flowing through them. If Analysts try to remember it all, then chances are for important elements to be left out. Therefore the information of the data flow should be recorded.

2. Communicate Meaning

Data Dictionary assists in ensuring common meanings for system elements and activities. It records additional details about the data flow in a system so that all persons involved can quickly look up the descriptions of data flows, data stores or processes.

3. Document System Features

Documenting the features of an information system is the third reason for using Data Dictionary systems. Features include the parts or components and the characteristics that distinguish each. Why each process is performed and how often it is used is documented.

Documenting system features produces more complete understanding. Once the features are articulated and recorded, all members of the project will have a common source for information about the system.

4. Facilitate Analysis

The next reason for Data Dictionary is to determine whether new features are needed in a system or whether changes of any type are in order.

5. Locate Errors And Omissions

The dictionary consists of information of transactions, inquiries, data, and capacity - this tells us a great deal about the system and allows you to evaluate it.

This information needs to be checked to ensure its completeness and accuracy. The dictionary is used to locate errors in the system descriptions such as :-

- Conflicting data flow descriptions

- Processes that neither receive input or generate output

- Data stores that are never updated

These need to be corrected. In dictionaries the process of recording the information will usually reveal errors.

2.5.3 What Does A Data Dictionary Record ?

All parts of an information system such as transactions, inquiries, reports, output files and databases depend upon data.

The dictionary contains two types of descriptions for the data flowing through the system :

* Data elements

* Data structures

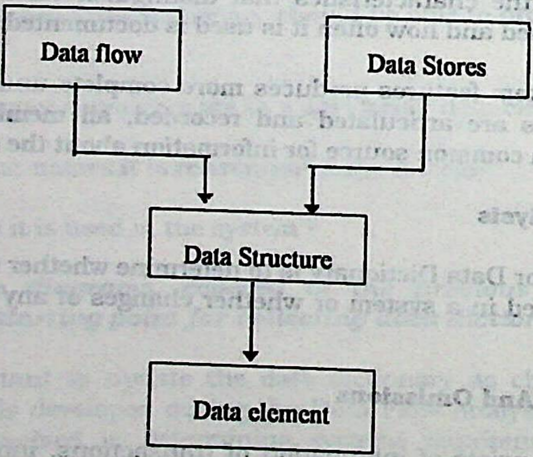


Figure 2.7 Data Description hierarchy

Data Elements

Data elements the most fundamental data level. They are grouped together to make up a data structure. Other names for data element are field, data item or elementary item. It is the smallest unit which has meaning. E.g. of data elements are employee name, grade, date of joining etc. These are grouped together to make up the employee register. They are building blocks for all other data in the system. By themselves they are meaningless to the user.

Data Structures

Data structure is a set of data elements that are related to one another and that collectively describe a component in the system. e.g. Of Data structure: employee register, consists of data elements such as employee name, date of joining, exit code, exit date, date of birth, department, grade and so on.

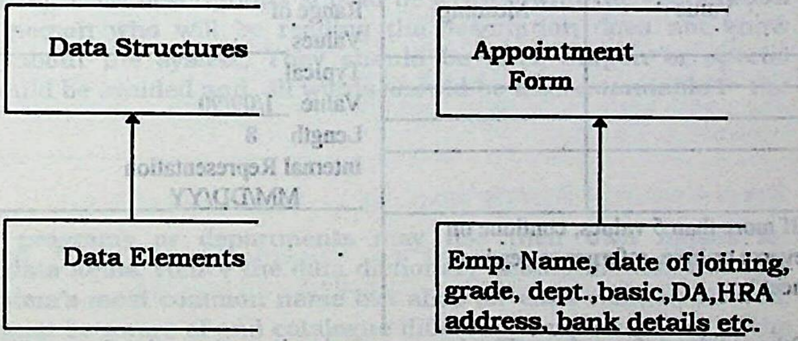

Data Structures
Data Elements
Appointment Form
Emp. Name, date of joining, grade, dept.,basic,DA,HRA address, bank details etc.

Figure 2.8 Relation of components in DFD

Both data flows and data stores are data structures. Another way of saying this is: *if data structures are moving, they are called data flows. when data structures are at rest and not moving they are data stores.*

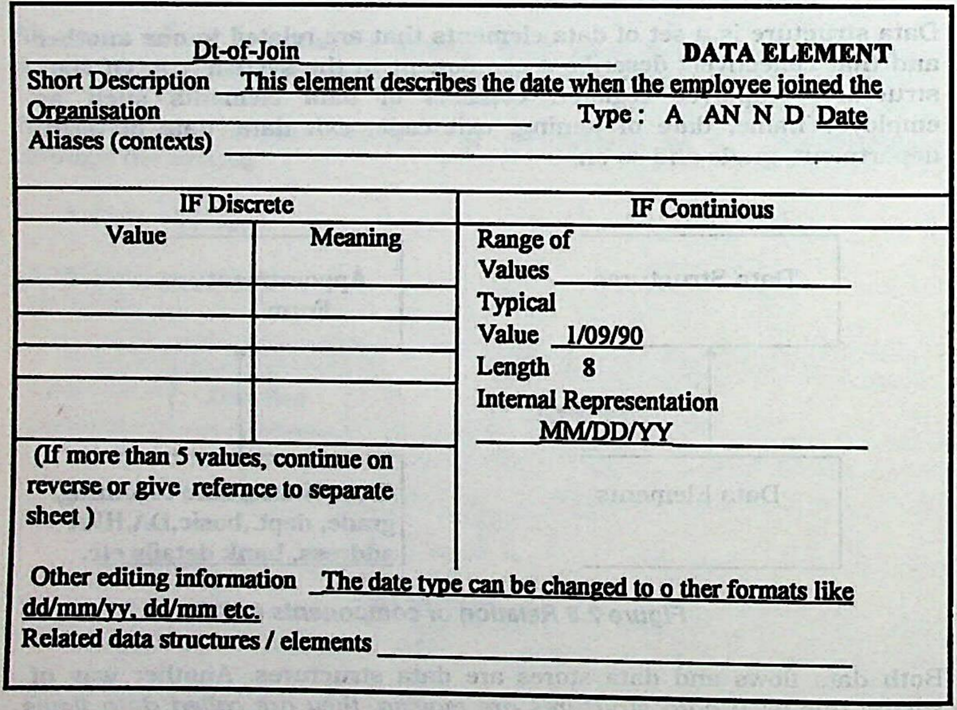
All data structures are defined in a dictionary entry. They consist of relevant elements that describe the activity or entity being studied.

The employee register data structure consists of the following major components. The data structures are broken down to their lowest level data items. For e.g. employee name, date of joining, exit code, exit data , department, grade, basic salary, DA, HRA, Bank code, bank name, bank address, account no., monthly net salary earned.

Describing Data Elements

All entries in the data dictionary consists of a set of details describing the data used or produced in the system. Each item is identified by a data name. They have a description, alias, length and has specific values that are allowed for it in the system being studied.

Figure 2.9 shows a specimen form for recording a data element.



Dt-of-Join **DATA ELEMENT**

Short Description This element describes the date when the employee joined the Organisation Type : A AN N D Date

Aliases (contexts)

IF Discrete		IF Continious
Value	Meaning	Range of Values
		Typical Value 1/09/90
		Length 8
		Internal Representation MM/DD/YY
(If more than 5 values, continue on reverse or give refernce to separate sheet)		

Other editing information The date type can be changed to o ther formats like dd/mm/yy, dd/mm etc.

Related data structures / elements

Fig 2.9 Specimen form for recording data elements

Data Names

To distinguish items of data from another, analysts assign meaningful names. The names are used to refer to each element throughout the entire systems development process. Hence it is advisable to select meaningful and understandable data names. E.g. the date of joining is more meaningful if it is named DATE OF JOINING rather than ABCXXX.

A common standard specifies that a data name should not exceed 30 characters (capital letters A to Z, numbers 0 - 9, and the Hyphen) and the data name should not contain any blanks. The date may then be written as DATE-OF-JOINING.

Data Descriptions

The Data Description briefly states what the data item represents in the system. e.g. The description of DATE-OF-JOINING indicates the date on which the employee joined the company (to distinguish this from his date of birth). Data descriptions should be written with the assumption that the person who will be reading the description does not know anything about the system. They should be brief, Jargon or special terms should be avoided and all words should be understandable to the reader.

Aliases

Different programs or departments may use their own names for common data items. Hence the data dictionary should include not only the data item's most common name but alias for each element as well. Analyst must be aware of and catalogue different terms that refer to the same data item. This helps to avoid duplication of effort. It allows better communication between organisational departments sharing a database and makes maintenance more straight forward.

Length

During Systems design process it is important to know the amount of space needed for each data item. These details can be captured during data flow analysis. Length identifies the number of spaces (for letters, numbers or symbols) required. It does not specify how they are stored. For example - if employee name can be upto 30 characters long when written in the register, the data dictionary entry should show a length of 30.

Data values

In some processes, only specific data values are allowed. For example the department code of the employee could be a one letter code. This may also include a range of values. This detail should be present in the data dictionary description of department - data item in the following manner :

Code	Department
1	Sales
2	Accounts
3	Administration

The system may later be designed so that the above codes are the only codes that are acceptable input.

Another example could be the sex of the employee where the only codes that could be accepted are 'M' or 'F' referring to male or female.

In case data values are restricted to a specific range, that information should be present in the data dictionary. For example the employee number should have a three digit identification number. These details are important to the analysts later, when they design systems controls. They will want to be sure that the system treats a four-digit employee number as an error.

Recording Data Descriptions

A complete explanation of all elements in the DFD includes a description of each data flow, data store and process

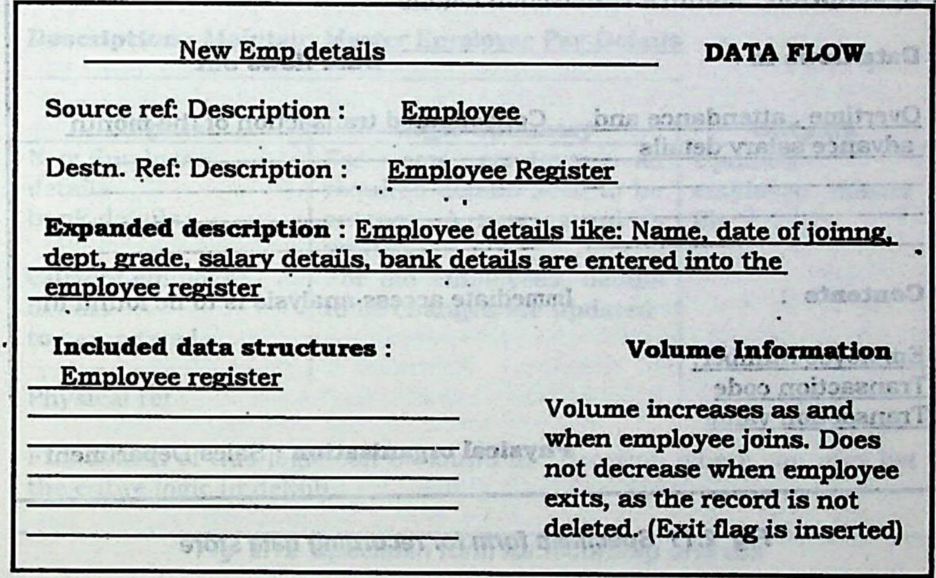
Describing Data Flows

Data flows are data structures in motion. The contents of a data flow are expressed by defining the names of the data structure that pass along it.

It consists of :

- The source of the data flow
- The destination
- The volumes of each data structure or transaction
- The present physical implementation of the data flow

A simple form of Data Flow is shown in figure 2.10



New Emp details **DATA FLOW**

Source ref: Description : Employee

Destn. Ref: Description : Employee Register

Expanded description : Employee details like: Name, date of joinng, dept, grade, salary details, bank details are entered into the employee register

Included data structures :

Employee register

Volume Information

Volume increases as and when employee joins. Does not decrease when employee exits, as the record is not deleted. (Exit flag is inserted)

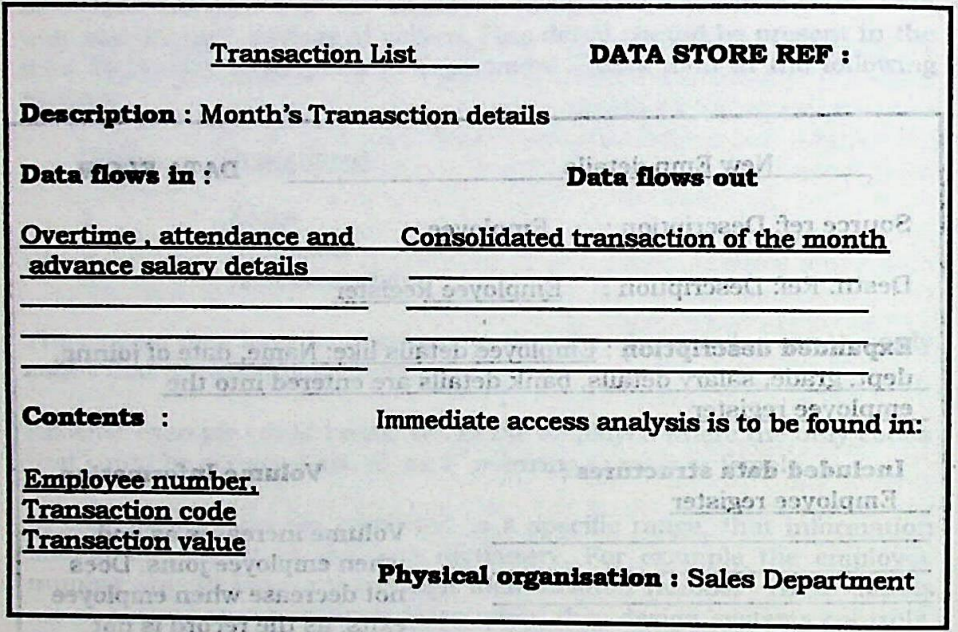
Fig 2.10 Specimen form for recording data flow

Fig 2.10

Describing Data Stores

As mentioned earlier a data store is a data structure at rest. The contents of each data store is described in terms of data structures found in it. It is also made up of the data flows that are input and those that are output from it. While describing the physical organisation of the data store, the details of primary key, secondary key should be included.

A simple form is shown in figure 2.11



Transaction List

DATA STORE REF :

Description : Month's Tranasction details

Data flows in :

Data flows out

Overtime , attendance and advance salary details	Consolidated transaction of the month

Contents :

Employee number,
Transaction code
Transaction value

Immediate access analysis is to be found in:

Physical organisation : Sales Department

Fig 2.11 Specimen form for recording data store

Describing Processes

The logic of processes are documented in various ways such as decision trees, decision tables, and structured English. The full detailed description of the logic of a process cannot be included in the data dictionary at all times.

What is included in the data dictionary to describe a process is:

- Inputs and outputs of the process
- Logic is summarised
- Reference to the place in the functional specification documentation where the logic is explained.

Figure 2.12 shows the specimen form for recording process.

Description : Maintain Master Employee Pay Details

Inputs	**Logic Summary**	**Outputs**
New Employee details bank details	For new employee, all required details need to be entered. A new record is written.	Updated employee master file
Current employee details to be updated	For old employees, details to be changed are updated	

Physical ref :

Full details of this logic can be found in : (location where you may list the entire logic in detail).

Fig 2.12 Specimen form for recording process

Figure 2.12

The above form should be filled keeping the following rules in mind:

- ☞ **The name of the process mentioned here should be the same as that in the DFD.**

- ☞ **The description of the process should be brief but clearly understandable by any person involved in the system.**

- ☞ **The logic summary, i.e. the centre column should state the main functions clearly and precisely.**

Describing Glossary Entries

In a number of applications, the user has his own jargon which the analyst or other team members may not be familiar with. The data dictionary is a convenient place to record these glossary items.

2.6 Introduction to Prototyping

The systems prototype method involves the user more directly in the analysis and design experience than does the systems development life cycle or structured analysis method. Prototyping is very effective under the correct circumstances. However, it is useful only if it is employed at the right time and in the appropriate manner.

What is A Prototype ?

A prototype is a working system - not just an idea on paper - that is developed to test ideas and assumptions about the new system. It consists of working software that accepts inputs, performs calculations, produces printed or displayed information, or performs other meaningful activities. It is the first version of an information system - an original model.

The design and the information produced by the system are evaluated by users. This can be effectively done only if the data are real and the situations live. Changes are expected as the system is used.

The underlying principle of prototyping is : Users can point to features they like or dislike and so indicate short comings in an existing and working system more easily than they can describe them in a theoretical or proposed system. Experience and use produce more meaningful comment than analysis of charts and narrative proposals.

Exercise - 2

1. Context diagram consists of ______ process, ______ and ______.
2. Write down the steps to draw a DFD
3. Give three reasons to state why a data dictionary is important
4. A Data dictionary contains two types of descriptions. What are they?

Ans : 1) one process, data flow, and entities
2) Make list of external entities, data flows; process & data stores
Draw the context level diagram

Fill in the details for steps within each process - levelling
Derive at a logical DFD

3) To manage details, To communicate the meaning the document system features, facilitate analysis, and locate error and omissions

4) Data elements and Data structures

Summary

Structured Analysis focuses on specifying what the system or application is required to do. Essential elements of structured analysis include graphic symbols, **data flow diagrams** and a centralised **data dictionary**.

Data flow analysis studies the use of data in each activity. It documents these findings in data flow diagrams, which graphically show the relation between processes and data, and in data dictionaries which formally describes the system data and where they are used.

- A physical data flow diagram shows actual devices, departments, people etc. Involved in the current system.
- A logical data flow diagram shows the proposed system.
- Drawing a data flow diagram involves:
  1. Developing the DFD using the top-down approach- context diagram.
  2. Fill in the details - levelling
  3. Deriving the logical DFD
- The Data flow diagram should be evaluated for correctness.
- A data dictionary is a catalogue of all elements in a system.
- The data dictionary is important
  - to manage the details
  - to communicate meaning
  - document system features
  - facilitate analysis

  - ✈ locate errors and omissions.

- A data dictionary records data elements and data structures.
- A prototype is a working system

✈ not just an idea on paper

✈ developed to test ideas and assumptions about the new system.

Chapter 3

Normalization

At the end of this session, you will be able to:

- ➔ *Have an idea of database design*
- ➔ *Define a relation*
- ➔ *Know the purpose for normalization*
- ➔ *Understand the steps involved for normalization*

3.1 Database Design

Having identified all the data in the system, it is necessary to arrive at the logical database design. Database Design involves designing the **conceptual model** of the database. This model is independent of the physical representation of data. Before actually implementing the database, the conceptual model is designed using various techniques.

The requirements of all the users are taken into account to decide the actual data that needs to be stored in the system. Once the conceptual model is designed, it can then be mapped to the DBMS/RDBMS that is actually being used. Two of the widely used approaches are Entity-Relationship(E/R) Modeling and Normalization.

The E/R model is an object-based model and is based on a perception of the real world that is made up of a collection of objects or **entities** and the **relationships** among these. E/R Modeling is generally used as a top-down approach for new systems.

Normalization is a technique that is more applicable to record-based data models for e.g. a relational database model. Each of the processes can be carried out independently to arrive at normalized tables(depending on how detailed the decomposition is). If E/R Modeling is done in detail, normalization may not be required at all. However, some people use the E/R diagram as an input to normalization i.e. if the tables derived from E/R diagrams are in the first normal form.

In this session we will concentrate on Normalization which is an important step in database design, particularly for relational DBMSs. The relational data model, is based on a *relation*. When structuring data that is to be stored, the analyst must anticipate the need to access the data to meet unexpected requirements, and to reduce redundancy. These can be achieved through the techniques of *'normalization'* that provides a systematic way of boiling data structures down to their simplest possible forms.

3.1.1 What Is A Relation?

A 'relation' is a two-dimensional table. It consists of 'rows' which represent records and columns which show the attributes of the entity. A relation is also called a file, it consists of a number of records which are also called a tuples. Records consists of a number of attributes which are also known as fields or domains.

*In order for a relational structure to be useful and manageable, the relation tables must first be **'normalized'**.*

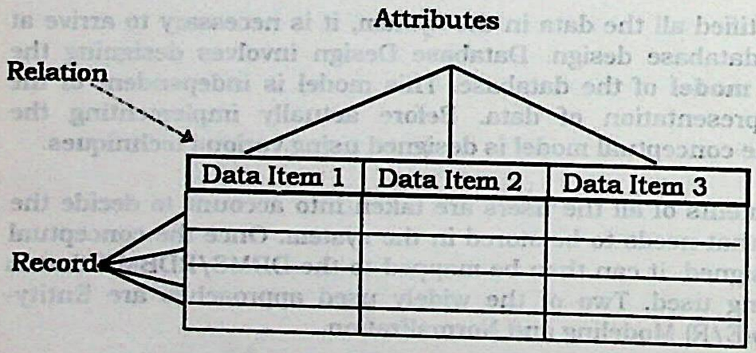


Figure 3.1 shows the components of a relation.

Some of the properties of a relation are

- **No duplication** : In the sense that no two records are identical.
- **Unique key**: Each relation has a unique key by which it can be accessed

- **Order** : There is no significant order of data in the table.

Figure 3.2 shows a relation (Unnormalized form) of the employee entity. In case we want the names of all the employees whose grade is 20, we can scan the employee relation, noting the grade. Here the *unique key* is the employee number.

3.2 What Is Normalization

Normalization is a process of simplifying the relationship between data elements in a record. It is the transformation of complex data stores to a set of smaller, stable data structures.

Normalized data structures are simpler, more stable and are easier to maintain. Normalization can therefore be defined as a process of simplifying the relationship between data elements in a record.

3.3 Purpose For Normalization

Normalization is carried out for the following four reasons :-

- To structure the data so that there is no repetition of data, this helps in saving space.

- To permit simple retrieval of data in response to query and report requests.

- To simplify the maintenance of the data through updates, insertions and deletions.

- To reduce the need to restructure or reorganize data when new application requirements arise.

Exercise - 1

1.Relation is a ____________ consisting of______and ________.

2. Normalized data structures are _______, more ______ and are ______ to maintain.

3. In order for a relational structure to be useful and manageable, the relation tables must first be ______________.

Ans: 1) simpler, stable and easier.
2) two-dimensional table,rows and columns
3)Normalized

3.4 Steps Of Normalization

Systems analysts should be familiar with the steps in normalization, since this process can improve the quality of design for an application. ***Starting with a data store developed for a data dictionary*** the analyst normalizes a data structure in three steps. Each step involves an important procedure to simplify the data structure.

It consists of basic three steps :

1. First Normal form which decomposes all data groups into two-dimensional records.
2. Second Normal Form which eliminates any relationships in which data elements do not fully depend on the primary key of the record.
3. Third Normal Form which eliminates any relationships that contain transitive dependencies.

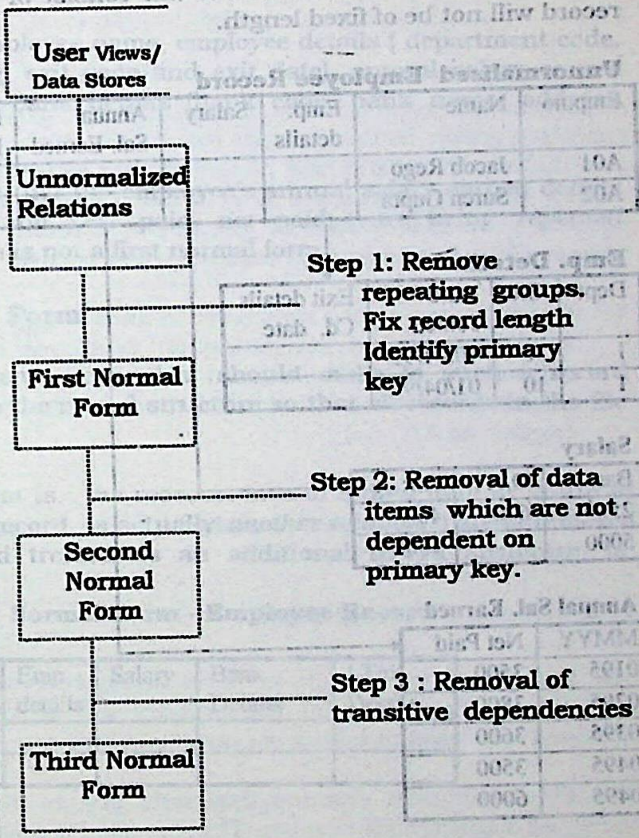
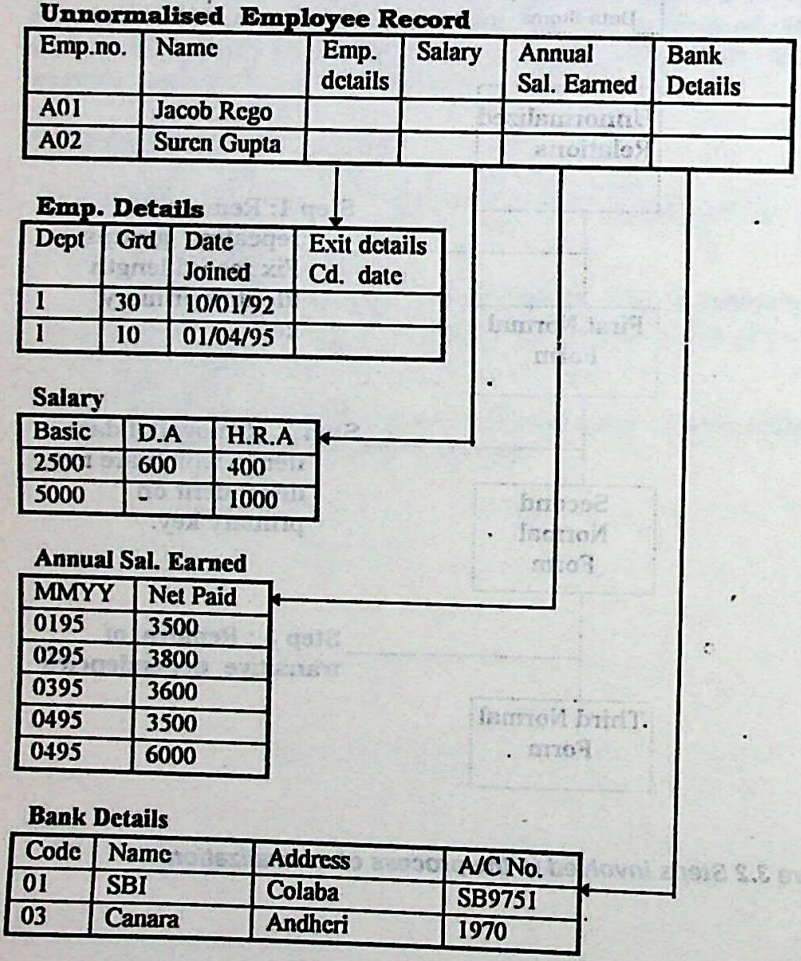

User Views/
Data Stores
Unnormalized Relations
Step 1: Remove repeating groups. Fix record length Identify primary key
First Normal Form
Step 2: Removal of data items which are not dependent on primary key.
Second Normal Form
Step 3 : Removal of transitive dependencies
Third Normal Form

Figure 3.2 Steps involved in the process of normalization.

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

The relation obtained from the data store such as Employee register will most likely be ***unnormalized.*** It will consist of repeating groups and record will not be of fixed length.



Unnormalised Employee Record

Emp.no.	Name	Emp. details	Salary	Annual Sal. Earned	Bank Details
A01	Jacob Rego				
A02	Suren Gupta				

Emp. Details

Dept	Grd	Date Joined	Exit details Cd. date
1	30	10/01/92	
1	10	01/04/95	

Salary

Basic	D.A	H.R.A
2500	600	400
5000	-	1000

Annual Sal. Earned

MMYY	Net Paid
0195	3500
0295	3800
0395	3600
0495	3500
0495	6000

Bank Details

Code	Name	Address	A/C No.
01	SBI	Colaba	SB9751
03	Canara	Andheri	1970

Figure 3.3 Unnormalised Employee Record

Figure 3.3 shows an unnormalized form of an employee record, this consists of:

Employee no., employee name, employee details (department code, grade, date of joining, exit code and exit date), annual salary earned (MMYY , net paid); bank details (bank code, bank name, address, employees A/C no) .

Here it is clearly seen that the employee's annual salary earned details which are : Month and Year paid, net paid, are being *repeated*. Therefore this relation is not a first normal form.

3.4.1 First Normal Form

The basic improvement the analyst should make to such a record structure is to design the record structure so that all records in the file are of fixed length.

A *repeating group,* that is, the reoccurrence of a data item or group of data items within a record, is actually *another relation*. This is removed from the record and treated as an additional record structure, or relation.

First Normal Form - Employee Record

Emp.no.	Name	Emp. details	Salary	Bank Details	I.Tax Details
A01	Jacob Rego				
A02	Suren Gupta				

Annual Sal. Earned record

Emp.no	MMYY	Net Paid
A01	0195	3500
A01	0295	3800
A01	0395	3600
A01	0495	3500
A02	0495	6000

Figure 3.4 First Normal Form

Figure 3.4 shows the normalization to first normal form for the employee record.

As mentioned above the first normal form is carried out by removing the repeating group. In this case we remove the Annual salary earned items and include them in a new file or relation called Annual Salary earned record. **Employee number is still the primary key in the employee record. A combination of employee number and MMYY is the primary key in the annual salary earned record.**

We thus form two record structures of fixed length:

> **Employee record** consisting of: Employee no., employee name, employee details (department code, grade, date of joining, exit code and exit date), bank details (bank code, bank name, address, employees A/C no) .

> **Annual salary earned record** consisting of - employee no., month & year(MMYY) and net paid.

3.4.2 Second Normal Form

The second normal form is achieved when every data item in a record that is *not* dependent on the primary key of the record should be removed and used to form a separate relation.

The PF department ensures that only one employee in the state is assigned a specific PF number. This is called a ***one-to-one relation***. The PF number uniquely identifies a specific employee; an employee is associated with one and only one PF number. Thus, if you know the Employee no., you can determine the PF number. This is ***functional dependency***. *Therefore a data item is functionally dependant if its value is uniquely associated with a specific data item.*

Now consider our Employee record example. The employee record in figure 3.4 is the first normal form.

Try the following test to check whether it is also a second normal form:

> In case the bank code is known, will you know the employee no.? In this case 'no', as a one-to-one relation does not exist. Because the bank name is dependent on bank code and not employee no, and because the relation between the primary key of bank code and

employee no. is not one-to-one, (it is not functionally dependent) we know that the second normal form has not been achieved. Therefore, an additional record structures called Bank record is created as shown in fig 3.5

Second Normal Form - Employee Record

Emp.no.	Name	Emp. details	Salary	A/C No.	Bank code	I.Tax Details
A01	Jacob Rego			SB9751	01	
A02	Suren Gupta			1970	03	

Annual Sal. Earned record

Emp.no	MMYY	Net Paid
A01	0195	3500
A01	0295	3800
A01	0395	3600
A01	0495	3500
A02	0495	6000

Bank Record

Code	Name	Address
01	SBI	Colaba
03	Canara	Andheri

Figure 3.5 Second Normal Form

The three record structures that are created are :

1. **Employee record** consisting of: Employee no., employee name, employee details (department code, grade, date of joining, exit code and exit date), bank details (bank code, bank name, address, employees A/C no) .

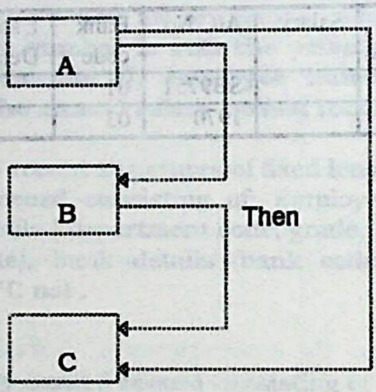
2. **Annual salary earned record** consisting of - employee no., month & year(MMYY) and net paid.

3. **Bank record** consisting of : bank code, bank name and bank address. All the attributes of this relation are ***fully dependent*** on Bank code.

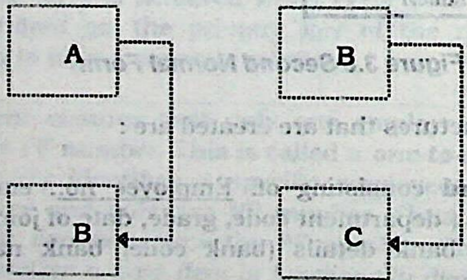
The primary key of each of the record structures are underlined.

3.4.3 Third Normal Form

Third normal form is achieved when transitive dependencies are removed from a record design. Some of the non-key attributes are dependent not only on the primary key but also on a non-key attribute. This is referred to as a *transitive dependency*.



Conversion to third normal form removes transitive dependence by splitting the relation into two relations.



Reason for concern. When there is a transitive dependence, deleting A will cause deletion of B and C as well.

Fig. 3.6 Third Normal Form

As per figure 3.6

- ➢ A, B and C are three data items in a record.
- ➢ If C is functionally dependent on B and
- ➢ B is functionally dependent on A,
- ➢ Then C is functionally dependent on A,
- ➢ Therefore, a transitive dependency exists.

The record in figure 3.5 is in second normal form. There are **no transitive dependencies**, so it is also in third normal form.

Exercise - 2

1. What are the steps of Normalization.
2. What is functional dependency.

Ans : 1) 1. Removal of repeating groups, fixing record length, identification of primary key.
2. Removal of data items which are not dependent on primary key
3. Removal of transitive dependencies.

2) a data item is functionally dependant if its value is uniquely associated with a specific data item.

Summary

In this chapter we have learnt the following:

- A ***relation*** is a two dimensional table, consisting of rows which represent records, and columns which represent attributes of the entity.
- ***Normalization*** is the process of simplifying the relationship between data elements in a record.

Normalization is carried out for four reasons which are :

✈ to structure the data

- to permit simplc retrieval of data in response to query
- to simplify the maintenance of data
- to reduce the need to restructure or reorganize data for new requirements.

The relation obtained from a data store developed for a data dictionary will most likely be '*unnormalized*'. The normalization process consists of three basic steps which are :

1. Removal of repeating groups, fixing record length, identification of primary key - **first normal form.**
2. Removal of data items which are not dependent on primary key - **second normal form**
3. Removal of transitive dependencies - **third normal form.**

Chapter 4

Process Specifications

At the end of this session, you will be able to :

- ➔ *Know Decision Concepts*
- ➔ *Understand Structured English and the three basic types of structured statements*
- ➔ *Understand the characteristics of Decision Trees*
- ➔ *Understand the characteristics of Decision Tables*
- ➔ *Build Decision Tables*
- ➔ *Identify the types of Table Entries*

4.0 Introduction

The logic of the system cannot be documented by any of the tools examined so far. The data dictionary contains the details of processes but, does not contain the logic used to convert inputs into outputs. The method of operation and logical procedures of each of the bottom level processes found in a *logical DFD* need to be examined and documented for study.

In this chapter, we will examine tools meant for this purpose. Tools help analysts assemble information gathered through the data collection methods discussed. This chapter covers the three tools used for documenting procedures and system logic. They are - Structured English, Decision Trees, and Decision Tables.

4.1 Overview

The Systems Analyst approaching structured decisions has many options for documenting and analyzing them. So far we have not dealt with the logic used in processes. The logic for each of the bottom level processes found in the logical DFD need to be examined.

The methods available for documenting and analyzing the logic of decisions (of a process) include :

- Structured English
- Decision Trees
- Decision Tables

Structured decisions consist of systematic methods that promote completeness, accuracy, and communication.

Decision analysis focuses on the logic of the decisions that are made, or need to be made, within the organization to carry out the objectives of the project.

Once all the process procedures are documented, the process procedures and logic should be reviewed by the users to ensure accuracy.

4.2 Decision Concepts

When analyzing procedures and decisions, the analyst must start by identifying conditions and actions. When you look at a system and ask,

What are the possibilities ?
or
What can happen ?

then you are looking at ***conditions.***

There are various conditions in a procedure. When all possible conditions are known, the analyst must now determine ***what*** to do ***when*** certain conditions occur.

Actions are alternatives - the steps, activities, or procedures that one may decide to take for a set of conditions. Actions could be simple or extensive. In many procedures, analysts must consider combinations of conditions and actions.

Process can be broken into:

- Sequence of actions
- Selection of actions based on some conditions
- Repetition of actions

Decision Trees, Decision Tables and Structured English are tools used to help understand and match combination of conditions and actions.

4.3 Structured English

As mentioned above there are three tools for decision analysis of structured decisions also called 'Pseudocode'. One of them being *Structured English*. This method is used when the decisions are not very complex. This method makes use of narrative statements to describe a procedure.

Structured English specifications require the analyst to identify :

- the conditions that occur in a process
- the decisions that must be made when these conditions occur
- Actions to be taken.

This method allows the analyst to list the steps in the order in which they should be taken. *No symbols or formats* are used. Entire procedures can be stated in English- like statements.

On the whole structured English consists of

- Structured logic or instructions organized into nested and grouped procedures
- Simple English statements such as add, multiply, move, and so on.

4.3.1 Developing Structured Statements

Structured English uses three basic types of statements to describe a process:

- Sequence Structures
- Decision Structures
- Iteration (repeating) Structures

These work well for decision analysis. They can be carried forward into programming and software development.

Statement forms of the three structures are shown in figure 4.1

Structured English Type	Example
SEQUENTIAL STRUCTURE A block of instructions where no branching occurs.	Action #1 Action #2 Action #3

DECISION STRUCTURE
Only IF a condition is true, complete the following statements otherwise jump to the ELSE
	IF condition A is true
	THEN implement Action A
	ELSE implement Action B

ITERATION STRUCTURE Blocks of statements that are repeated until done	DO WHILE there are employees Action #1 ENDDO

Figure 4.1

◆ Sequence Structures

This includes a block of instructions where no branching occurs. In other words they are a set of *actions* without the existence of *conditions*. Typically, several sequence instructions are used together to describe a process.

Example :
Process to update a particular employee record as he has resigned.

1. Get particular employee record
2. Enter '1' in Exit code data element
3. Enter date of resigning in exit date.
4. End of job

This simple example shows a sequence of four steps. Note that none of the steps contain a decision or any condition that determines whether the steps are taken.

The steps are carried out in *order*. ***In sequence structures, steps are always carried out, one after the other***. The four sequence steps are always carried out one after the other and without any decision about order or exceptions.

◆ Decision Structures

Decision structures are used when two or more actions can be taken, depending on the value of a specific condition. The condition should be assessed and then the decision is to be made and the set of actions for that decision should be carried out. Once the condition is determined the actions are unconditional.

The decision structure uses the phrases IF-THEN-ELSE to point out alternatives in the decision process. Decision Structure can have many condition-action combinations .

The following example shows the nesting of multiple levels of conditions and actions for each decision point.

```
If employee does not exist
   Else
      If grade < 20
      DA = 20% of basic
      HRA = 0
   Else
       If grade < 30
       DA = 20% basic limit to 600
       HRA = 400
       Else
      If grade < 40
      DA = 0
      HRA = 40% of basic
      Else
            DA = 0
           HRA = 50% of basic salary
End if
```

As can be seen above there are a number of conditions and action combinations. The example shows the nesting of multiple levels of conditions and actions for each decision point. IF-THEN-ELSE are the phrases used in the example. This clearly tells us the logic for the process of DA and HRA calculation.

◆ Iteration Structures

It is common to find activities which

- Have to be repeated *'while'* a certain condition exists
- OR repeated *'until'* a particular condition occurs.

Iteration instructions describe these cases.

Example:

```
  DO WHILE there are employees
  If employee does not exist
     Else
       If grade < 20
       DA = 20% of basic
       HRA = 0
     Else
        If grade < 30
        DA = 20% basic limit to 600
        HRA = 400
        Else
       If grade < 40
       DA = 0
       HRA = 40% of basic
       Else
            DA = 0
           HRA = 50% of basic salary
  End if
  End do
```

In the above example we have used DO WHILE. You can use DO UNTIL as well. In both cases the process will be repeated till the condition specified exists.

4.4 Decision Trees

Decision Trees are used when the condition and action combination is not very complex. Trees are also useful when it is essential to keep a string of decisions in a particular sequence.

4.4.1 Decision Tree Characteristics

A Decision tree is a diagram that presents conditions and actions sequentially (in order) thus showing the order of conditions. This method shows the relationship of each condition and its permissible actions. The diagram resembles branches of trees, hence the name.

Figure 4.2 shows a decision tree where the sequence of decisions is from left to right. The root is on the left, this is the starting point of the decision sequence, the branching moves towards the right. The particular branch to be followed depends on the conditions that exist and the decision to be made.

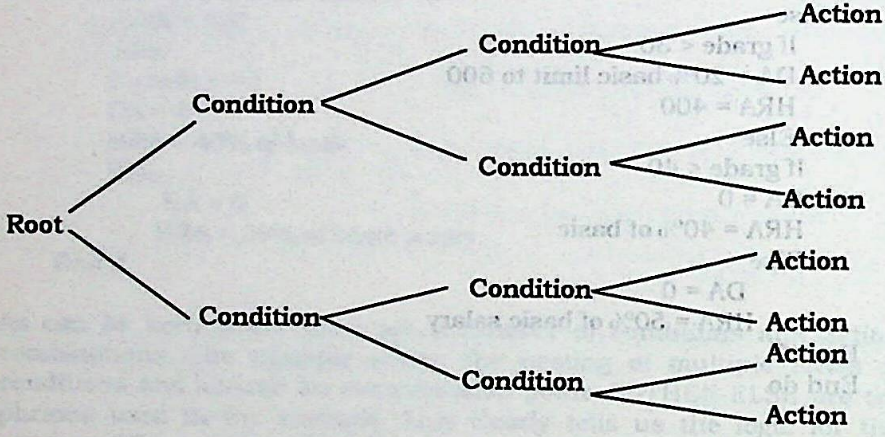


Fig. 4.2

Movement is from left to right, along a particular branch and is the result of making a series of decisions. Each node of the tree represents a condition. Before moving to the next path, a decision on which condition exists *has* to be made. The right side of the tree lists the actions to be taken, depending on the sequence of conditions that have been followed.

4.4.2 Using Decision Trees

One of the benefits of using a decision tree is that analysts are forced to formally identify the actual decisions that must be made. Connecting decision trees into an If-Else structure is very easy when many conditions have to be checked for.

It is not easy for them to overlook an integral step in the decision process. Decision trees force analyst to consider the sequence of conditions.

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

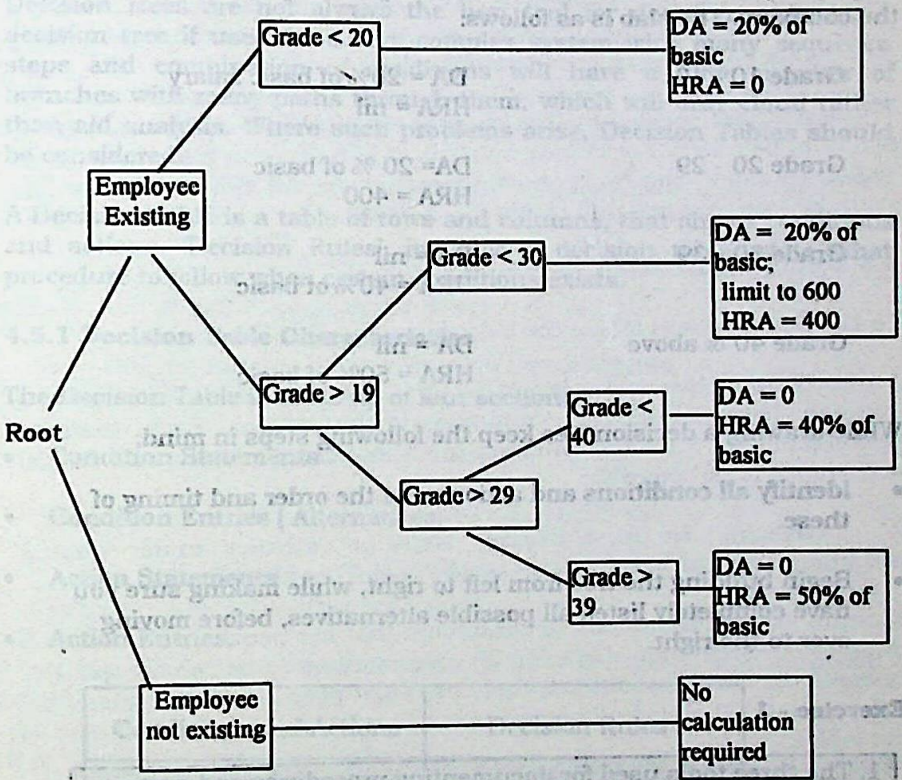


Figure 4.3

These points can be clearly seen in figure 4.3 which shows the process to arrive at the DA and HRA of an employee, based on the grade of the employee. Calculation should be done only for employees still existing is the company. The slab is as follows:

Grade 10 - 19	DA = 20% of basic salary
	HRA = nil

Grade 20 - 29	DA= 20 % of basic
	HRA = 400

Grade 30 - 39	DA = nil
	HRA = 40% of basic

Grade 40 & above	DA = nil
	HRA = 50% of basic

While drawing a decision tree keep the following steps in mind:

- Identify all conditions and actions and the order and timing of these.
- Begin building the tree from left to right, while making sure you have completely listed all possible alternatives, before moving over to the right.

Exercise - 1

1. The three tools used for documenting procedures and system logic are ____________, ________________ and ______________
2. Structured English uses three basic types of statements to describe a process. Name them.
3. In sequence structures, steps are always carried out,___________.
4. In a decision tree movement is always from ______________ along a particular branch.

Ans: 1) a) Structured English b) Decision Trees c) decision tables
2) a) sequence structures b) decision structures c) Iteration structures
3) one after the other
4) right to left

4.5 Decision Tables

Decision trees are not always the best tool for decision analysis. A decision tree if used for a very complex system with many sequence steps and combination of conditions will have a large number of branches with many paths through them, which will only cloud rather than aid analysis. Where such problems arise, Decision Tables should be considered.

A Decision Table is a table of rows and columns, that shows conditions and actions. 'Decision Rules', included a decision table, state what procedure to follow when certain conditions exists.

4.5.1 Decision Table Characteristics

The Decision Table is made up of four sections:

- Condition Statements
- Condition Entries (Alternatives)
- Action Statements
- Action Entries.

Conditions and Actions	Decision Rules
Condition Statements Entries(Alternatives)	Condition
Action Statements	Action Entries

Fig 4.4

The *condition statement* identifies the relevant conditions.

Condition Entries (alternatives) tell which value, if any, applies for a particular condition.

Action Statement list the set of all steps that can be taken when a certain condition occurs.

Action Entries show what specific actions in a set to take, when selected conditions or combinations of conditions are true.

You can enter a note below the table to help state when to use the table or to distinguish it from other tables. The columns on the right linking conditions and actions form decision rules.

These state the conditions that must be satisfied for a particular a set of action to be taken. For Decision Tables unlike decision trees there is no order sequence. The decision rule incorporates 'all' the conditions that must be true, not just one condition at a time.

4.5.2 Building Decision Tables

To Develop Decision tables the following steps should be used:

1. Determine the most relevant factors to be considered in making a decision. This identifies the conditions in the decision. Each condition selected should have the potential to either occur or not occur. Partial occurrence is not possible.

2. Determine the most feasible steps or activities under varying conditions (not just the current conditions). This identifies the actions.

3. Study the combinations of conditions that are possible. For every N number of conditions, there are 2^N combinations to be considered. For example, for three conditions, there are eight possible combinations; $2^3 = 8$. For four, $2^4 = 16$. combinations are possible and can be included in the table.

4. Fill in the table with decision rules. There are two ways to fill in the table.

   1. In a longer method, here you have to fill in condition rows with a yes or no value for each possible combination of conditions.
   2. The other method of completing the table deals with one condition at a time and adds to the table for each additional condition but does not add duplicate combinations of conditions and actions, as discussed.

      a. State the first condition and permissible actions.

      b. Add the second condition by duplicating the first half of the matrix and filling in the different Y and N values from the new condition in both halves of the expanded matrix.

c. Repeat set b for each additional condition.

5. Mark action entries with X to signal action(s) to take; leave cells blank or mark with a dash to show that no action applies to that row.

6. Examine the table for redundant rules or for contradictions within rules(discussed below).

Following the above simple guidelines will help to :

- save time in building a decision table from collected information.
- point out where information is missing
- show where conditions do not matter in a process
- indicate where there are important relations or results that others were not aware of in other words not considered

Thus using decision tables can produce more complete and accurate analysis.

4.5.3 Checking Decision Tables

After constructing a table analysts verify it for correctness and completeness to ensure that the table includes all the conditions, along with the decision rules that relate them to actions. Analysts should also examine the table for redundancy and contradictions.

Eliminating Redundancy

Decision Tables are likely to get too large if allowed to grow in an uncontrolled way. Removing redundant entries help to manage table size. Redundancy occurs when:

1. Two decision rules are identical except for one condition row

2. The actions for two rules are identical.

- **Removing Contradictions**

Decision rules contradict each other when two or more rules have the same set of conditions and the actions are different. This could occur when either there is an error in constructing the table or when the analysts receive discrepant information from different individuals about how decisions are made.

- **Impossible Situations**

When building decision tables, it is possible to set up impossible situations. Now let us see an example for impossible situations. One thing we have to make sure is to see that accuracy is maintained and redundancy is avoided. In the table 1 below, Rule 1 is not possible as a person who is earning more than Rs. 50,000 per year cannot earn less than Rs. 2000 per month at the same time.

Conditions and Actions	Rules			
	1	2	3	4
Salary > Rs. 50,000 per year	Y	Y	N	N
Salary < 2000 per month	Y	N	Y	N
Action 1				
Action 2				

Table 1 - Decision table checking for impossible situations

Impossible situation

Developing Decision Tables

In order to build decision tables, the analyst needs to determine the maximum size of table, eliminate any impossible situations, inconsistencies or redundancies, and simplify the table as much as possible.

1. Determine the number of conditions that may affect the decision. Combine the rows that overlap. For example, conditions that are mutually exclusive. The number of conditions becomes the number of rows in the top half of the decision table.

2. **Determine the number of possible actions that can be taken. This becomes the number of rows in the lower half of the decision table.**

3. **Determine the number of conditions alternatives for each condition(rules).**

4. **Calculate the maximum number of columns in the decision table by multiplying the number of alternatives for each condition.**

5. **Fill in the condition alternatives. Start with the first condition and divide the number of columns by the number of alternatives for that condition. For e.g., for 2 conditions, there are 8 rules. Divide 8 by 2. (2^3). So write Y in four columns and N in remaining 4 columns as follows:**

Condition 1: Y Y Y Y N N N N
Condition 2: Y Y N N Y Y N N
Condition 3: Y N Y N Y N Y N

According to the decision tree example let us see the decision table given below:

Conditions and Actions	**Rules**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Grade between 10-19	Y	Y	Y	Y	N	N	N	N
Grade between 20-29	Y	Y	N	N	Y	Y	N	N
Grade between 30-39	Y	N	Y	N	Y	N	Y	N
Actions								

Note: It is assumed that all employees exist.

6. **Complete the table by inserting an X where rules suggest certain actions.**

7. **Combine rules where it is apparent that an alternative does not make a difference in the outcome. For e.g.,**

Condition 1:	**Y**	**Y**
Condition 2:	**Y**	**N**
Action 1	**X**	**X**

This can be expressed as:

Condition 1:	**Y**
Condition 1:	**-**
Action 1	**X**

The dash (-) signifies that condition 2 can be either Y or N and the action will still be taken.

8. **Check the table for any impossible situations, contradictions, and any redundancies. According to the earlier example there are certain impossible situations and redundancies. Take rule 1, 2, 3 and 5. In all these rules, there are two Y's, actually speaking a person cannot have two grades, he has to be in a particular grade, so these rules have to be eliminated as, it is an impossible situation.**

   The last rule says that a employee exists but then there is no grade given to that employee, here there is a contradiction so again this has to be eliminated.

9. **Rearrange the conditions and actions (or even rules) if this makes the decision table more understandable.**

Note: **The decision table for the example above has been explained in the limited entry table after eliminating all impossible situations.**

4.5.4 Type Of Table Entries

There are four types of table forms:

- **Limited-Entry Form**
- **Extended-Entry Form**
- **Mixed-Entry Form**
- **ELSE form**

☞ Limited-Entry Form

The basic table structure consists only of 'Y', 'N', and blank entries. This is a limited-entry form. It is the most commonly used formats.

This decision table is made according to the example given in decision tree. We have arrived at this table after eliminating all the impossible situations, contradictions and redundancies. This has already been explained.

Decision Table with Limited Entry form

Conditions and Actions	Rules				
	1	2	3	4	5
Employee exists	Y	Y	Y	Y	N
Grade between 10-19	Y	N	N	N	-
Grade between 20-29	N	Y	N	N	-
Grade between 30-39	N	N	Y	N	-
Grade greater than 39	N	N	N	Y	-
DA 20 % of Basic, HRA= 600	X				
DA 20 % of Basic , HRA = 400		X			
DA = 0, HRA = 40 % of Basic			X		
DA = 0, HRA = 50% of Basic				X	
No Calculation					X

Table 2

In a limited entry decision table, the condition are expressed as simple Yes or No questions, whereas in a Extended Entry table conditions have more than two possible states.

☞ Extended-Entry Form

The extended-entry form replaces Y and N with action entries telling the reader how to decide. Here the condition and action statements themselves are not complete, therefore the entries contain more than one Yes and No.

In the extended entry form, the condition is that, if an employee exists, and if he is in marketing department ("MKT") he is eligible for commission. The commission is based on the total sales made for the month and is calculated as follows:

If sales >=10000 then commission is 20%
if sales >=5000 and <10000 then commission is 10%
if sales <5000, commission is 0

The table is shown in table 3.

Conditions and Actions	Rules				
	1	2	3	4	5
Employee exists	Y	Y	Y	Y	N
Department	MKT	MKT	ADM	MKT	-
Total sales	10,000	2000	-	5000	-
Commission applicable	20%	0	n.a.	10%	n.a

Table 3

Note: n.a is Not Applicable

The decision table above explains how the commission is given according to the sales per month. According to these conditions the actions have to be taken. In the first rule, the person exists, he is in "MKT" department, and his total sales is upto 10000, so he is eligible for commission, he will get commission, 20% of sales amount.

In rule 3, the person is in "ADM" department, so he does not have any sales figure, so he is not eligible for any commission as such.

☞ Mixed-Entry Form

This form consists of combined features of limited and extended-entry forms. Generally only one form should be used in each section of the table, but between the condition and action sections, either form may be used.

Now let us see an example of the mixed entry form, where there are various combination of conditions and actions taken.

According to the company's rules and policies, following are the conditions and actions to be taken

1) **For all employee in the grade 10-19 and those in marketing department commision is calculated as follows:**

If sales >=10000 then commission is 20%

if sales >=5000 and <10000 then commission is 10%

if sales <5000, commission is 0

If the person is in any other department then he is not entitled for commission.

2) **Second condition is to check if the employee is supposed to pay income tax. This is done as follows:**

If the employee's salary >= 50,000 - Tax applicable

if salary <50000 - Tax not applicable

Tax is calculated for employees of all departments.

Conditions and Actions	Rules				
	1	2	3	4	5
Department	MKT	MKT	FIN	FIN	MKT
Grade 10- 19	Y	Y	Y	Y	Y
Salary >= 50,000 p.a.	Y	N	Y	N	N
Salary < 50,000 p.a.	-	Y	N	Y	Y
Total sales	20000	10000	-	-	3000
Commission applicable	20%	10%	n.a	n.a	0
Income Tax applicable	X	-	X	-	-
Income Tax exempted	-	X	-	X	X

Table 4

The decision table in table 4 explains the following :

In the first rule, the person is in "MKT" department, the grade is between 10 and 19, and the salary per annum is more than 50,000 and sales is Rs. 20,000. The action taken for this rule is - 20% commission is given because he is in marketing department and he is applicable for Income tax.

In rule 4, the employee is in Finance ("FIN")department, grade is between 10 -19, and salary less than 50,000 per annum, so he is exempted from income tax, and he is not given any commission as he is not in marketing department.

In rule 5, he is in marketing department, his grade is between 10-19, he need not pay income tax as salary less than 50,000, he is not given commission as his sales is only Rs. 3000.

All these tables are made while design specifications are made. Each table is made according to a required module, and you can see how each table is designed and how actions are taken.

☞ **ELSE form**

This form aims at omitting repetition by using ELSE rules.
To build an ELSE form decision table

- specify the rules with condition entries to cover all sets of actions except for one
- This will be the rule to follow when none of the other explicit conditions are true.
- This rule is the final column on the right, the ELSE column.
- If none of the other conditions are true, then the ELSE decision rule is followed.

The ELSE rule eliminates the need to repeat conditions that lead to the same actions.

The conditions for the ELSE form is that the employees of marketing department are given commissions according to the total sales done.

The commission is given as follows:

If Sales >= 10000 then 20% commission
if sales >=8000 and sales<10000 then 15% commission
if sales >=5000 and sales <8000 then 10% commission
else
if sales <5000 then 2% commission

Conditions and Actions	Rules			
	1	2	3	4
Sales >= 10000	N	Y	N	E L S E
Sales >=8000 and <10000	Y	-	N	
Sales >=5000 and <8000	N	-	Y	
Commission	15%	20%	10%	2%

Table 5

The Else form decision table shown below has an extra ELSE column. The ELSE is applicable for all marketing department employees who have made sales less than 5000.

The first rule tells us that the employee is in marketing department, his sales amount is between 8000 and less than 10000 so he gets 15% commission.

Exercise -2

1. What are the four sections in Decision tables.
2. Name the four types of decision tables
3. In a limited-entry form the basic table structure consists only of ________.
4. What does an ELSE form used for?

Ans : 1) Condition statements, Condition entries, Action statement , Action entries
2) Limited-Entry Form   Extended-Entry Form Mixed-Entry Form
ELSE form
3) 'Y', 'N', and blank entries.
4) This form aims at omitting repetition by using ELSE rules.

Summary

Documenting organization decisions and processes requires the identification of *conditions* and *actions* and learning about what information is available to suggest actions to take when specific combinations of conditions arise.

The three documentation tools for decision making and procedures are:

1. Structured English

2. Decision Trees

3. Decision Tables

Structured English **is used to state decision rules. The three types of statements are**

- **Sequence structures - shows unconditional actions.**
- **Decision structures - shows repetitive actions.**
- **Iteration structures - shows actions that occur only when and while certain conditions occur.**

Decision Trees **are presentations of decision variables that are graphic and sequential, showing which conditions to consider first, which second, and so on. The root of a decision tree is the starting point for analyzing a specific situation, and the branches indicate the sequence of decisions leading upto the proper action to take.**

Decision Tables **relates conditions and actions through decision rules. A decision rule states the condition that must be satisfied for a particular set of actions to be taken. The decision rule incorporates all the conditions that must be true at one time, not just one condition.**

There are four forms of decision tables:

- **Limited-entry**
- **Extended-entry**
- **Mixed-entry**
- **ELSE forms**

All forms should be developed without *redundancy* and *contradiction* .

Chapter 5

Structure Charts

At the end of this session, you will be able to:

➔ *Know what is Structured Design*

➔ *Understand what Structured Charts are and their purpose*

➔ *Define a Module*

➔ *Draw Structured Charts*

➔ *Know the principles of Structured Design*

➔ *Know Structured Flowcharts*

➔ *Understand Transaction Analysis*

➔ *Understand Transform Analysis*

5.0 Introduction

Structured Design which is one of the phases of System Development Life Cycle focuses on the development of software specifications. Structure Charts, show the relation of processing modules in computer software. This section discusses the development and use of structure charts. You will also be introduced to structured flowcharts, transaction analysis and transform analysis.

5.1 What Is Structured Design ?

Having completed the documentation of procedure and system logic, we can say that the Analysis stage has been completed. Structured Design uses Logical DFDs, Data dictionaries, normalized structures and process specifications which have been developed during analysis phase.

Structured Design, is another element that uses graphic descriptions. It focuses on software specifications. Well Structured designs improve the maintainability of a system. A *structured system* has the following qualities:

- It is developed from top-down
- It is modular, that is broken down into manageable components.
- The modules should be designed so that they have minimal effect on other modules of the system.
- Connections between modules are limited and interaction of data is minimal.
- This helps easing maintenance tasks.

The major tool used in Structured Design to depict the structure of a system is the ***structure chart.***

5.2 Structure Charts

The fundamental tool of Structured Design is the structure chart. It is a widely used tool for designing a modular, top-down system. The logical DFD forms the basis for drawing structure charts. Like Data Flow Diagrams, Structure Charts are graphic descriptions, they describe interaction between modules and the data passing between modules that interact with one another. These functional module specifications can in turn be passed to programmers prior to writing program code.

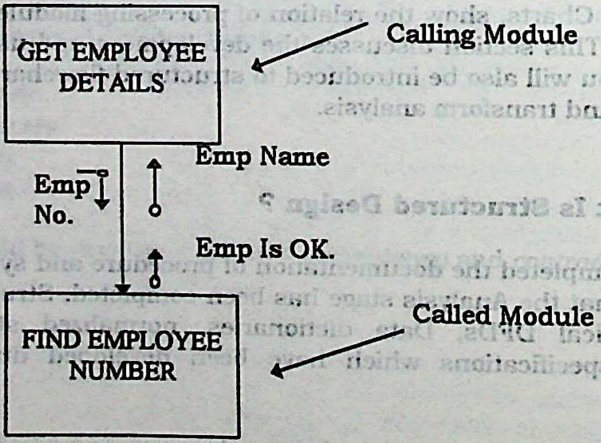


Figure 5.1 shows a structure chart

In the figure 5.1, 'GET EMPLOYEE DETAILS' calls 'FIND EMPLOYEE NAME'. There is data passing between the two modules.

- **GET EMPOLYEE DETAILS sends data emp. no. to FIND EMPLOYEE NAME.**

- **FIND EMPLOYEE NAME (having done its function) returns data Emp. Name to GET EMPLOYEE DETAILS, and**

- **FIND EMPLOYEE NAME also returns a flag (Emp. No. Is OK) to GET EMPLOYEE DETAILS . This is used to tell the calling module (caller) that everything went well because sometimes GET EMPLOYEE DETAILS may send a flag saying 'invalid emp. No.'.**

5.3 Purpose Of Structure Charts

A Structure Chart is a design tool that *visually displays the relationships between program modules*. It shows which modules within a system interact and also graphically depicts the data that are communicated between various modules as seen in Figure 5.1.

Structure Charts are developed prior to the writing of program code. They are not intended to express procedural logic (this is done by flowcharts and pseudocode), nor do they describe the actual physical interface between processing functions.

Structure Charts identify the data passes existing between individual modules that interact with one another. The communication between modules of a Structure Chart can be clearly seen in Figure 5.1.

5.4 Notation

A common notation is used in the development of structure charts for uniformity and ease of communication among systems developers.

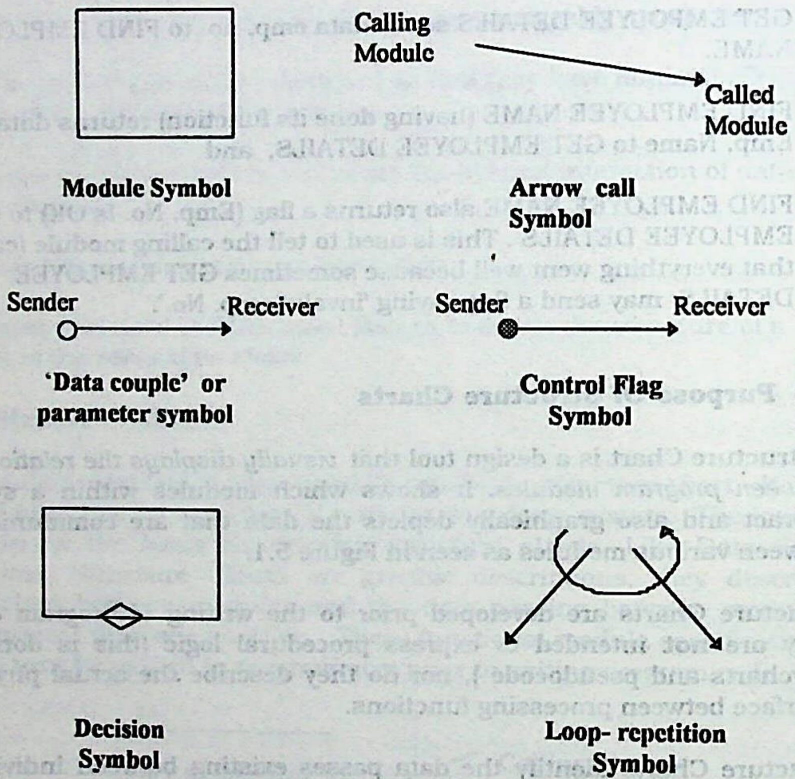


Figure 5.2 shows Primary Symbols Used in structure charts.

1. *Rectangles* : Represent modules. Module name is written inside the rectangle.

2. *Arrows* : Indicates that one module calls another; direction of the arrows indicates which module is calling. Arrow also implies transfer of information between modules. In traditional programming languages, call are made by the execution of PERFORM and DO statements.

3. *Small arrows with empty circles*: Is used to note the passing of data parameters - *'data couples'*. These are items of data needed in the called

module to perform the necessary work. They indicate that something is passed down to the lower module or back upto the upper one.

4. *Small arrows with filled in circles*: These represent control flags. Its purpose is to assist in the control of processing by indicating the occurrence of, say, end-of-file conditions or errors such as a transaction was not valid, or no such employee exists.

The fewer data couples and control flags one has in the system the easier it is to change the system. When these modules are actually programmed, it is important to pass least number of data couples or parameters between modules.

5. *Loop* : Indicates that the procedures found below this loop are to be repeated until finished.

Figure 5.2 (a) shows that the process 'CALCULATE EARNINGS AND DEDUCTIONS' found below the loop will be repeated until the net pay for all employees is completed.

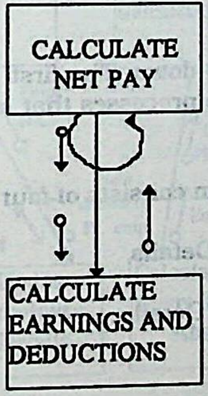


Figure 5.2(a)

6. *Small diamond* : This appears on the bottom of a rectangle. This signifies that only one of the modules below the diamond will be performed. In figure 5.3 the small diamond is found on the bottom of rectangle marked 'Maintain Master Employee Pay Details' . This indicates that either the module 'Write New Employee Details' or 'Update Employee Details' will be performed. This symbol represents the selection construct.

5.5 Definition Of A Module

In Structured Design, a *'module'* is defined as a set of instructions which can be invoked by name. It is a group of instructions, i.e., a paragraph, block, subprogram, subroutine or the like.

Contents of a module can be referred to collectively under a single label. The label should tell us just what the module does nothing more and nothing less.

Module names are important part of the notation. The rule is that the name of a module summarizes what the module does. The module name consist of an *Action verb* and *object noun* e.g. In the module called "CALCULATE NET PAY", CALCULATE is the action verb and NET PAY is the object noun. The name itself tells us that this module calculates the net pay but, it does not tell us how it does it. A module ideally serves only one function.

5.6 Drawing A Structure Chart

Structure Charts are to be drawn top-down. The first level logical DFD (figure 2.5) should be used to find the processes that are to become the modules in the structure chart.

Our first level logical Data flow diagram consists of four processes:

- 1. Maintain Master Employee Pay Details
- 2. Maintain Employee Transaction Details
- 3.Calculate Net Pay
- 4. Print Reports

These form modules in the structure chart.

STRUCTURE CHART

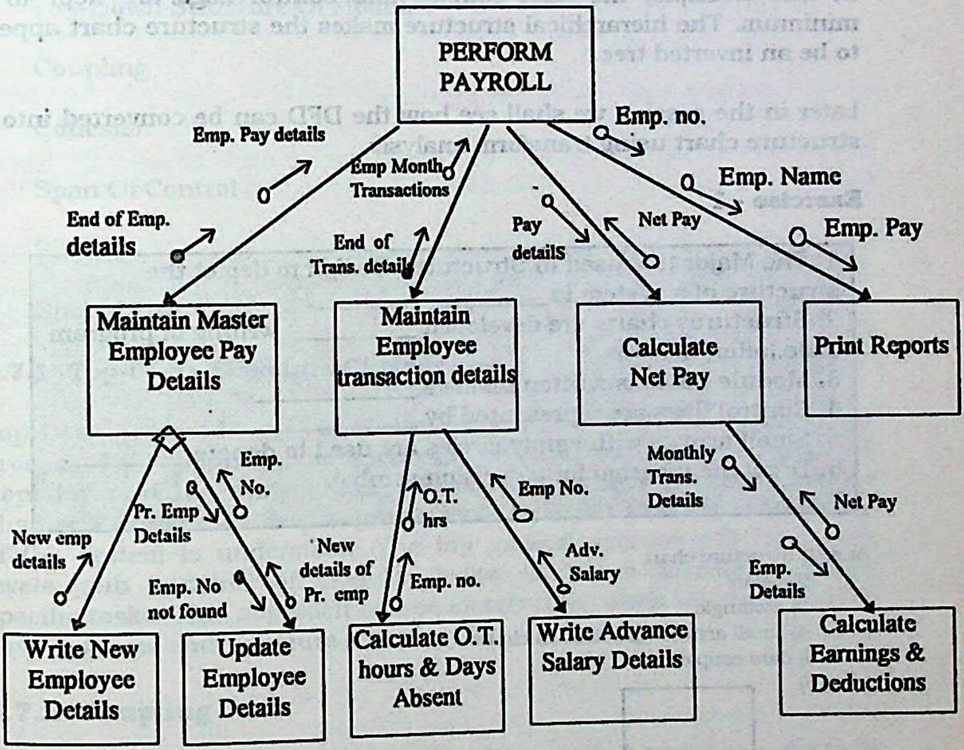


Fig. 5.3 Structure Chart

The Structure chart is shown in figure 5.3. The main process at the top of the chart is called "PERFORM PAYROLL" and represents the module that controls everything underneath. ***The modules on the second level bear similar names and as that of the processes in the data flow diagram..***

Since the data flow diagram is intended to be a logical representation of the system, it is not unusual that the modules would be the same. The modules on the second level will control the operations of the modules on the third level.

These modules accomplish separate functions such as "WRITE NEW EMPLOYEE DETAILS" , "CALCULATE EARNINGS AND DEDUCTIONS" and so on.

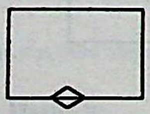
In this example, the data couples and control flags are kept to a minimum. The hierarchical structure makes the structure chart appear to be an inverted tree.

Later in the session we shall see how the DFD can be converted into a structure chart using transform analysis.

Exercise - 1

1. The Major tool used in Structured Design to depict the structure of a system is ______________.
2. Structures charts are developed __________ writing of program code.(after, before)
3. Module name is written inside a ___________
4. Control flags are represented by _______________________.
5. Small arrows with empty circles are used to denote__________.
6. Draw the diagram for a decision symbol.

Ans: 1) Structure chart
2) before
3) rectangle
4) small arrows with filled in circles
5) data couples
6)



5.7 Principles Of Structured Design

As already studied structure charts is the major tool used for structured design. Besides this there are other aspects that one should know of structured design.

In this section we will describe in brief each of the following points which are useful for structured design.

- Top-Down Structure Of Modules
- Coupling
- Cohesion
- Span Of Control
- Size
- Shared Use

5.7.1 Top-Down Structure Of Modules

Top-Down methods are used throughout the analysis and design process. This approach, starts with the understanding at a very general (top) layer of the system which is then *'exploded'* (as explained in chapter 2 under data flow diagrams) to lower levels where greater detail of the system is understood. The top general process will have sub levels (sub systems) of modules below it. These modules perform specific tasks. This approach can be clearly seen when we studied data flow diagrams and structure charts.

5.7.2 Coupling

'Coupling' refers to the 'strength' of the relationship **between** modules in a system. The strength of a relationship i.e., coupling is determined by the data passes between modules and on the dependence of one module on another for input. For a good system design, the dependence of one module with another should be minimum. This is called **loosely coupled**. When modules are loosely coupled, changes to one module will not have a cascading effect .

As mentioned above loose couplings minimizes the interdependence between modules. This can be achieved in the following ways :-

- **Control the number of parameters passed between modules.**
- **Avoid passing unnecessary data to called modules.**
- **Pass data (upward or downward) only when needed.**
- **Maintain superior/subordinate relationship between calling and called modules.**
- **Pass data, not control information.**

If you see figure 5.1 notice that employee number is sent by 'GET EMPLOYEE DETAILS' to 'FIND EMPLOYEE NUMBER'.

This employee number being the record key, distinguishes the employee record and helps to get all the required employee details. As it is the record key, it is unlikely to change, other items in the record may change. Hence, the loosely coupled alternative is better suited to achieving the stated design and maintenance objectives.

Passing too little data can make it impossible to perform the task(function). In case the employee number is not passed to the sub-ordinate module then it is not possible for the sub-ordinate module to know which record to locate.

***'Floating data'* should be avoided in designs. This occurs when one module produces data that are not required by the calling module but by another, elsewhere in the system. These details pass through the system till they reach the function that requires them.**

5.7.3 Cohesion

As seen in the previous section an important way to evaluate the partitioning of a system is by how cleanly the modules are separated from one another - that is the criterion of coupling.

Another way to determine partitioning is to look at how the activities 'within' a single module are related to one another. This is the criterion of *'cohesion'*.

A module should be highly '*cohesive*'; that is, each module should accomplish one and only one function. In properly modularized, cohesive systems, the contents of the module are so designed that they perform a specific function and are more easily understood by people than systems designed by other methods. Hence cohesive modules are not largely dependent on other modules.

Contents of the modules determine how cohesive the module is. These contents can be categorized in the following manner :

1. *Module contents not closely related :*

Modules in this case are developed by size or number of instructions i.e., when programmer works according to strict rules and divides modules into sections of 50 statements each; all modules must fit on a single page.

2. *Module contents determined by logic of processing:*

All steps are performed together or handle the same functions. Here the elements are related by 'time' at which they are performed; i.e. they logically seem to go together and are performed at the same time. For instance a module that initializes all variables and opens files is logically bound.

In this case all the elements are executable at one time. Modules that are logically bound are difficult to modify. Even the simplest change can effect all types of transactions. It is better to separate each type of transaction into its own module.

3. *Module Determined By Data Used:*

Here all elements in the module refer to the same data or files. A module that reads the next transaction and updates the master file by adding, deleting or changing records, including error checking required for each type of function, shares a common set of data. This type of binding is so far the best. e.g. Printing, displaying and copying data from a common file.

4. *Module Contents Determined by Function Performed:*

All activities in a module have the same single purpose, that is, perform a single function. This provides more thorough testing of the module. If changes are needed at a later time, one can easily determine how the

module is constructed and how it processes data and interacts with other modules in the system.

The emphasis on reliability and maintainability is constant throughout systems development.

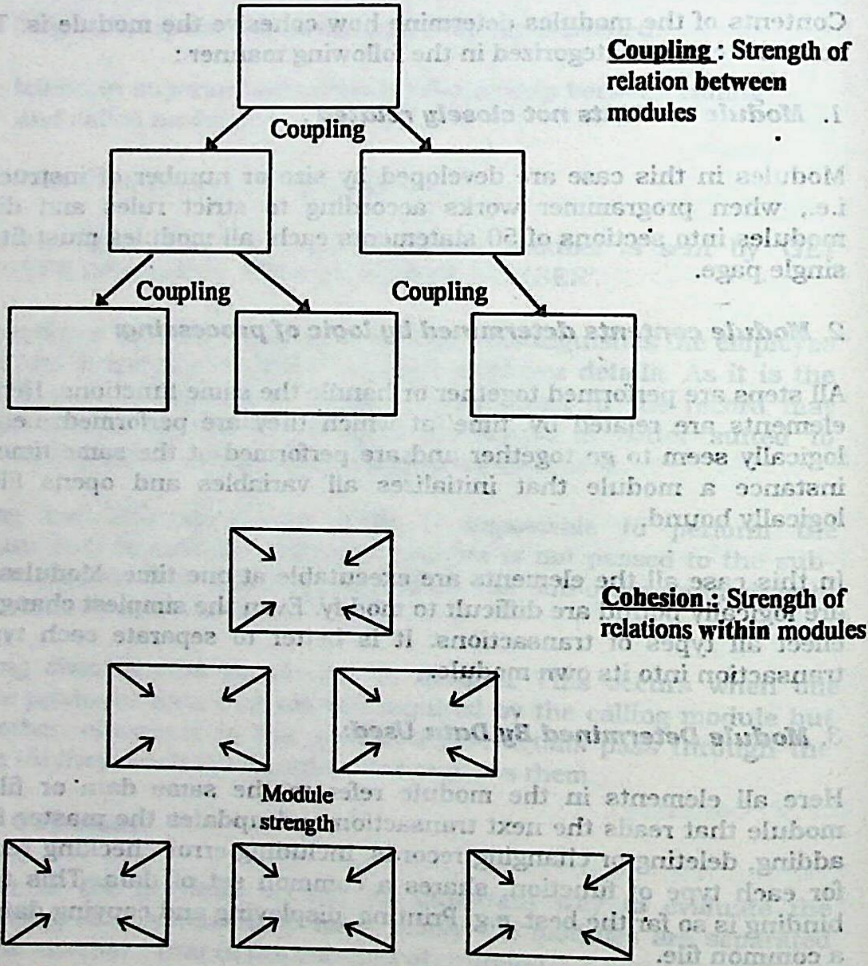

Coupling : Strength of relation between modules
Coupling
Coupling
Coupling
Cohesion : Strength of relations within modules
Module strength

Figure 5.4 depicts coupling and cohesion in software design.

Thus a good design should have an ideal combination of highly cohesive and loosely coupled modules.

5.7.4 Span Of Control

Span of control refers to the number of sub-ordinate modules controlled by a calling module. In general the number of sub-ordinate modules should be limited to, five to seven. Modules should interact with each other and manage the functions of a limited number of lower-level modules.

When the number of sub-ordinate modules are very high, then it gets difficult in determining which module to invoke under certain conditions. It is also difficult in establishing calling sequences to pass data and receive results.

This results from not applying objectives of coupling and cohesion.

5.7.5 Module Size

The number of instructions contained in a module should be limited so that module size is generally small.

Some of the guidelines (these should not be strictly used as thumb rules) to manage module size are :

- Module should contain not more than 50 instructions.
- Listing of source code for a module should fit on a single printed page.

In general modules should focus on a single purpose, be highly cohesive and loosely coupled. Yet the modules should not be too small. The size of the module also depends on the language. For example: COBOL code to perform a series of arithmetic calculations will be more than 'C' code OR COBOL code to generate a report will be more than FoxPro code.

5.7.6 Shared Modules

Functions such as calculations or a particular type of processing, *should not be* duplicated in separate modules. They should be established in a single module that can be invoked by any other module when needed. This minimizes the amount of software that must be designed and written. It also minimizes the number of changes that must be made during systems maintenance. For instance, if O.T calculation procedures change, only one module must be modified under the shared module principle (also known as library modules).

Many systems establish **library modules** - which are predefined procedures that are included in the system's program library. The routine is quickly invoked by a single command or call.

5.8 Structured Flowcharts

Structured flowcharts, also called Nassi-Schneiderman charts (or N-S charts), are graphic tools that force the designer to structure software that is both modular and top-down. The advantages of the Nassi-Schneiderman chart are :

- it adopts the philosophy of structured programming
- it uses a limited no. of symbols so that the flowchart takes up less space.

They provide a structure that can be retained by programmers who develop the application software. The responsibility of leveloping module logic, varies from organization to organization. The responsibility could be the analysts or the programmer's.

5.8.1 Basic Elements

There are three basic elements used in developing structured flowcharts. F:gure 5.5 shows the three basic symbols used to draw Nassi- Schneiderman charts.

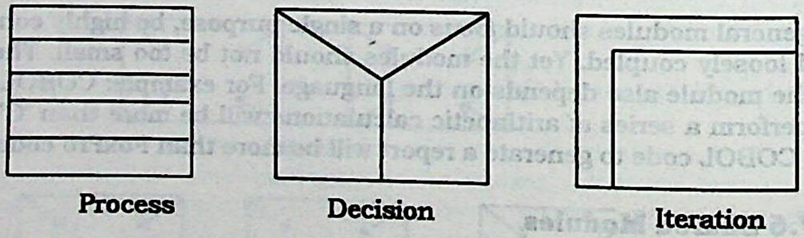


Fig. 5.5

There are many similarities between these elements and the components used in structured English.

Process: Simple processes or steps in a program are represented by a rectangular box, the process symbol. This symbol represents initialization of values, input and output activities, and calls to execute other procedures.

A name or brief description written in the box states the purpose of the process. The succession of steps is shown using several process boxes.

Decision : The decision symbol represents alternative conditions that can occur and that the program must have a manner of handling. Any form of a decision including several condition alternatives can be depicted using this symbol. They show the equivalent of the IF-*THEN*-ELSE structures discussed in structured English and common in many programming languages.

Iteration : The iteration symbol represents looping and repetition of operations *while* a certain condition exists or *until* a condition exists.

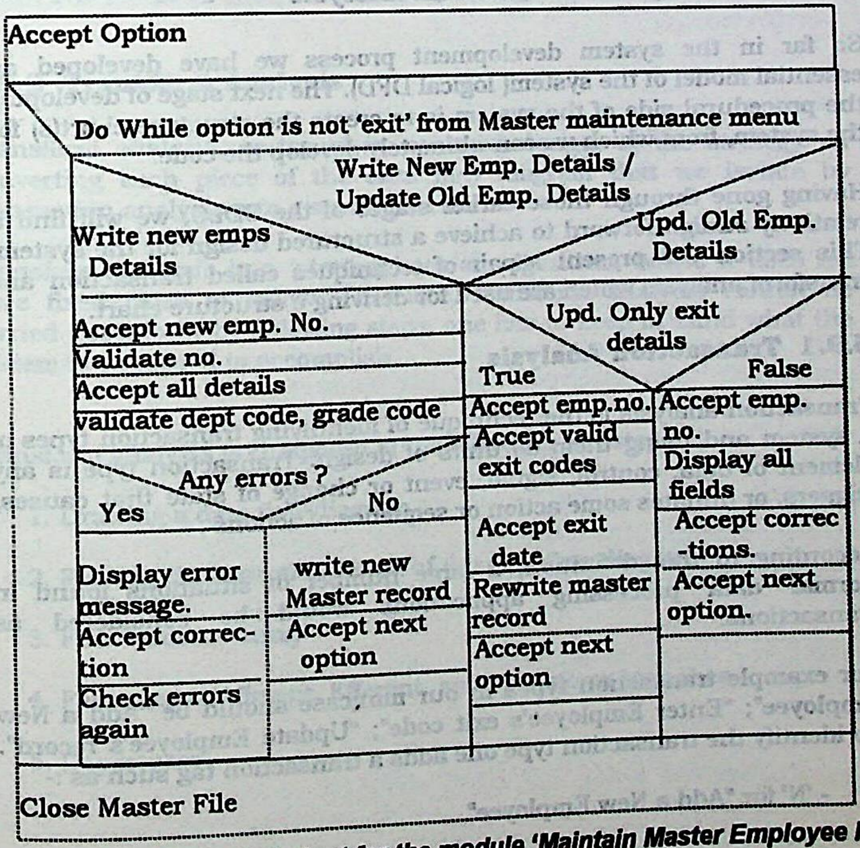


Figure 5.5 Sample N-S chart for the module 'Maintain Master Employee Pay Details'

The benefits of Nassi-Schneiderman charts are :

- ➢ They provide analysts with a tool for aiding the program design and development process because they are compatible with structured programming.

- ➢ This chart is easy to read because no knowledge of complex symbols is required.

- ➢ It does not take up precious space either.

- ➢ In summary, the Nassi-Schneiderman chart is a very valuable tool for the analyst.

5.9 Transaction And Transform Analysis

So far in the system development process we have developed an essential model of the system(logical DFD). The next stage of developing the procedural side of the system is to create the structure chart(s) for the system, from which we can ultimately develop the code.

Having gone through those earlier stages of the SDLC, we will find it relatively straightforward to achieve a structured design for the system. This section will present a pair of techniques called transaction and transform analysis which are used for deriving a structure chart.

5.9.1 Transaction Analysis

Transaction analysis is the technique of identifying transaction types of a system and using them as units of design. Transaction type is any element of data, control, signal, event or change of state that causes, triggers, or initiates some action or sequence of actions .

According to this definition, a large number of situations found in normal data processing applications would be considered as transactions.

For example transaction types in our minicase should be "Add a New Employee"; "Enter Employee's exit code"; "Update Employee's record". To identify the transaction type one adds a transaction tag such as :-

- 'N' for "Add a New Employee"

- 'E' for "Enter Employee's exit code"

- 'U' for "Update Employee's record".

The system needs some way to distinguish the transaction types, and the human (operator/user) needs some brief way to tell the system which type of transaction stimulus he is placing on the interface. This is done by the human user by selecting the 'code' for transaction type from a menu of transaction codes.

Figure 5.6 shows one transaction type of a payroll system that prints payslips for employees using variable data.

Having identified the transaction types we need to design each one separately, by the techniques of transform analysis.

5.9.2 Transform Analysis

Transform analysis, or transform-centered design, is the strategy for converting each piece of the data flow diagram that we isolate by transaction analysis into a structure chart.

Transform analysis is a *'strategy'* not an algorithm, hence it does not have fixed rules to follow. Therefore, transform analysis cannot be carried out blindly by following steps, one has to keep in mind what the system is supposed to accomplish.

Transform analysis is composed of the following five steps:

1. Drawing a data flow diagram

2. Finding the central function of the data flow diagram

3. First level factoring

4. Factoring of Afferent, Efferent, and Transform branches

5. Departures

1. Drawing a data flow diagram

In order to carry out the strategy of transform analysis, it is necessary to have a data flow diagram of the system's analysis. This as mentioned previously helps one to study the flow of data through a system.

2. Finding The Central Function Of The Data Flow Diagram

Some modules receive information from sub-ordinates, and then pass it upward to their super-ordinate. This is referred to as an afferent flow of data, and the modules which behave just opposite to afferent modules are **efferent modules.** These ***efferent modules take information from their super-ordinates and pass it down to their sub-ordinates.***

Central functions of a data flow diagram are those functions in the middle of afferent and efferent data elements.

Thus Afferent modules are those which are closest to the process which will transform them into the processes output, they are involved in the processing of the input.

While Efferent modules are those that maybe regarded as "logical output data" , those involved in the processing of the output.

Central functions is usually left in the middle after the afferent and efferent functions are identified. Central functions are the main work of the system. *They transform the major inputs into major outputs.*

Sometimes the afferent and efferent data elements are the same, in such cases there are no central functions.

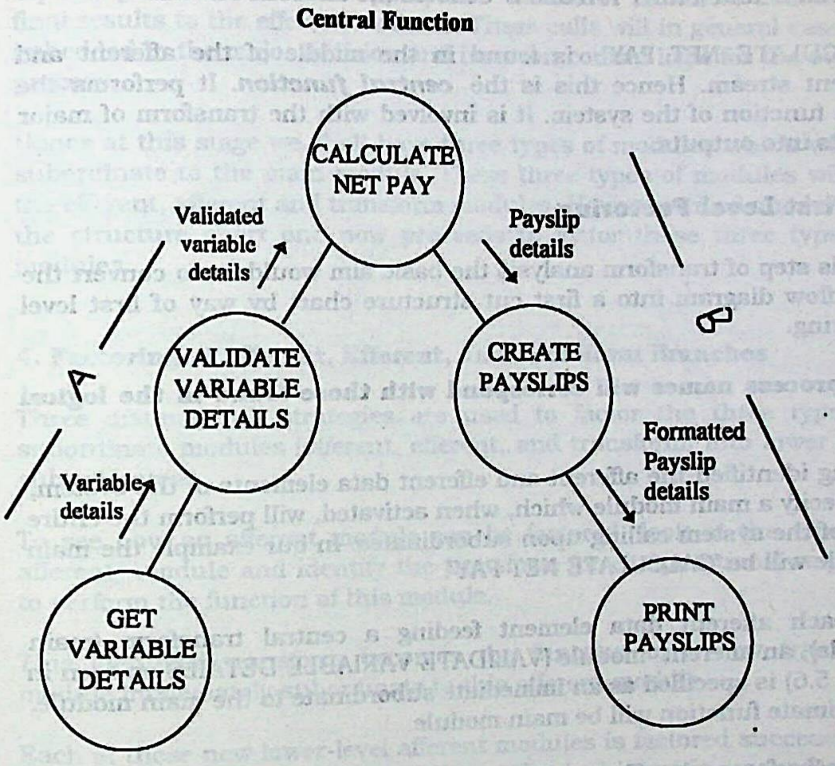


Figure 5.6

Figure 5.6 clearly shows that:

'VALIDATE VARIABLE DETAILS' receives information i.e. 'variables details' from its sub-ordinate 'GET VARIABLE DETAILS' and passes information i.e. 'validated variable details' upwards to its super-ordinate module called 'CALCULATE NET PAY'.

Thus the region marked 'A' which covers the modules 'VALIDATE VARIABLE DETAILS' and 'GET VARIABLE DETAILS' denotes the **afferent stream.**

In contrast, the module 'CREATE PAYSLIPS' takes information i.e. 'Payslip details' from its super-ordinate and passes it down as 'formatted payslip details' to its sub-ordinate 'PRINT PAYSLIPS'. These are involved in the processing of the output i.e. payslips.

Thus the region marked 'B' which covers the modules 'CREATE PAYSLIPS' and 'PRINT PAYSLIPS' denotes the **efferent stream.**

'CALCULATE NET PAY' is found in the middle of the afferent and efferent stream. Hence this is the ***central function.*** It performs the main function of the system. It is involved with the transform of major inputs into outputs.

3. First Level Factoring

In this step of transform analysis the basic aim would be to convert the data flow diagram into a first cut structure chart by way of first level factoring.

The process names will correspond with those found in the logical DFD.

Having identified the afferent and efferent data elements of the system, we specify a main module which, when activated, will perform the entire task of the system calling upon subordinates. In our example the main module will be 'CALCULATE NET PAY'.

For each afferent data element feeding a central transform (main module), an afferent module (VALIDATE VARIABLE DETAILS as seen in figure 5.6) is specified as an immediate subordinate to the main module. Its ultimate function will be main module.

Similarly, for each efferent data element emerging from any central transform, we define a subordinate efferent module (CREATE PAYSLIPS and PRINT PAYSLIPS as seen in figure 5.6) that will accept the efferent data element and, ultimately, transform it into the final physical output.

Finally for each central transform or functionally cohesive composition of central transform module, which will accept from the main module the appropriate input data and transform it into the appropriate output data; of course, this output is delivered back upward to the main module.

Thus, we can see that there is a simple correspondence between data flow diagram and the initial first level structure chart.

The main module is the overall control, or executive for the process. Its function is to control and coordinate the afferent, transform, and efferent modules dealing with the highest-level data of the system. It will call the immediately subordinate afferent modules to obtain major inputs, pass these to the appropriate transform modules, deliver the final results to the efferent modules. These calls will in general case, be imbedded in the major decision and iteration control logic for the overall process.

Hence at this stage we shall have three types of modules which will be subordinate to the main module. These three types of modules will be the efferent, afferent and transform modules. Having derived one level of the structure chart one now proceeds to factor these three types of modules.

4. Factoring of Afferent, Efferent, and Transform Branches

Three distinct sub strategies are used to factor the three types of subordinate modules (afferent, efferent, and transform) into lower level subordinates.

To see how an afferent module can be factored, look at the top-level afferent module and identify the transform (or computations) required to perform the function of this module.

This identified transform becomes the function of a new transform module immediately subordinate to this afferent module.

Each of these new lower-level afferent modules is factored successively, in the same manner until the ultimate physical input is reached or the process is otherwise terminated.

The factoring of efferent modules is essentially symmetrical to that of afferent modules. For a given afferent module, we are looking for the next transform to be applied which will bring the data closer to its ultimate "physical" form. The transform module that accomplishes this transformation will be subordinate to the "top-level" efferent module in the system.

For each transform, we look for a sub-transform that will compose the overall transform. We also look for compositions of the functions shown as the central transforms in the original data flow graph. These are inserted as intermediate modules in the hierarchy between the top-level and the functions from which they are composed.

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

After factoring of efferent, afferent and central branches is done one arrives with a structure chart as shown in fig 5.7.

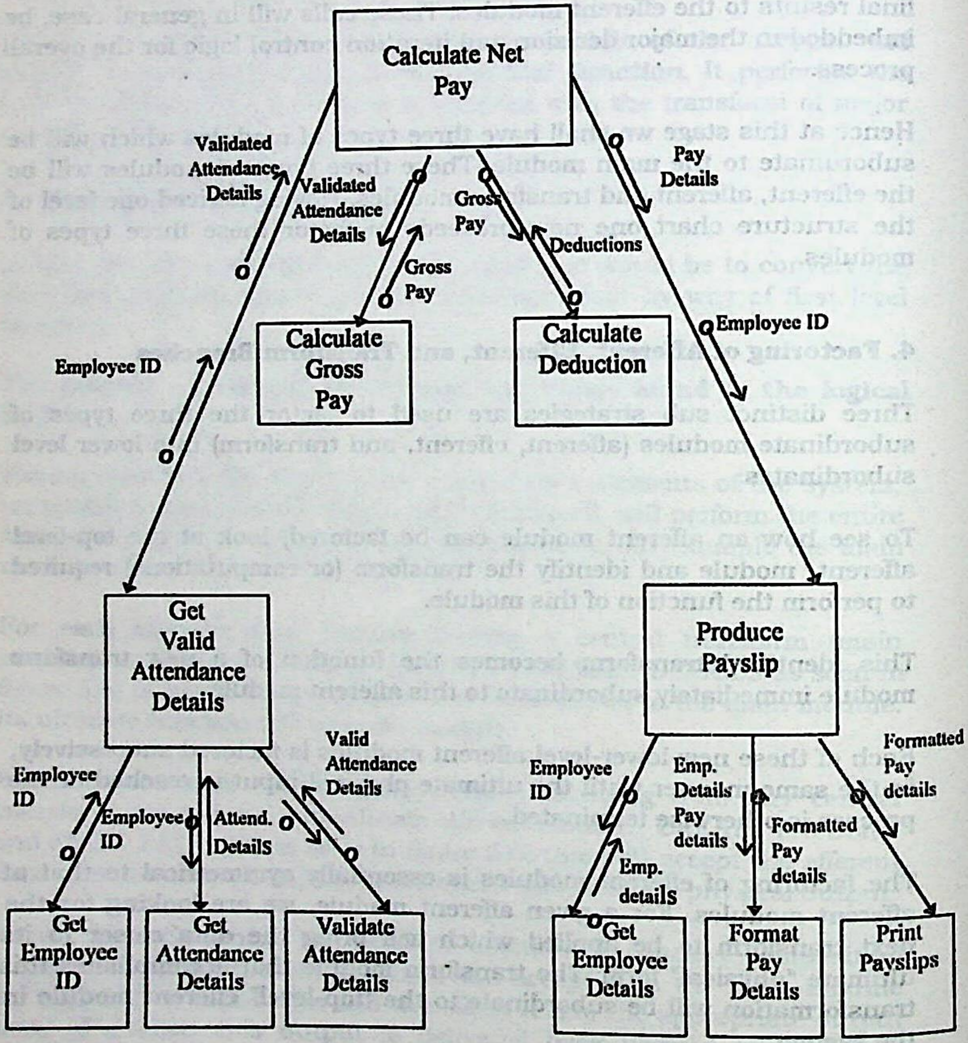


Fig. 5.7

5. Final step: Departures

This strategy described so far assumes an orthodox structure in which the data will flow inward or outward in any branch, but not both. Consequently, we expect that afferent modules will have only afferent and transform subordinates; similarly, efferent modules are expected to have only efferent and transform subordinates; and transform modules regardless of where they are in the structure should have only transform subordinates.

However, real world problems frequently require exceptions to these rules if clumsy processing is to be avoided. For example, we could require afferent subordinates to a transform module or we could require afferent subordinates to an efferent module.

We must always keep in mind that our objective is to make the program structure reflect the structure of the problem as closely as possible. The detailed data flow diagram is a guide to the problem structure , and if the problem requirements necessitate a departure from orthodox transform centered organization, it should be apparent in the diagram.

When completed, the trial structural design using transform analysis will bear a simple, straightforward relationship to the data flow. The final structure which will reflect many design trade-offs and heuristics, will be derived from systematic refinements and alterations of initial first-cut structure. These modifications may be made in a separate phase after completing the initial factoring, or as many designers prefer during the initial factoring.

Exercise - 2

1. __________ refers to the strength between the modules in a system.
2. __________ looks into how the activities within a single module are related to one another.
3. Give two rules to manage the module size.
4. ________ is the technique of identifying transaction types of a system using them as units of design.
5. ________ , is the strategy for converting each piece of the data flow diagram into a structure chart.

Ans: 1) Coupling
2) Cohesion
3) Module should contain not more than 50 instructions and listing of source code of a module should fit on a single printed page.
4) Transaction analysis
5) Transform analysis or transform-centered design

Summary

- **Structure charts** are an important tool for the software designer. They visually display the relation between program modules and graphically show the data communicated between each module.

  In general, two types of data are transmitted.

  1. Parameter data consist of those items needed by the module to do the necessary processing.

  2. Control information assists in the control of processing by indicating the occurrence of errors or conditions that affect process, such as the end-of-file.

- Six principles characterize good software designs:

  1. Top-down partitioning

  2. Loose coupling

  3. Functional grouping for cohesion

  4. Ltimited span of control

  5. Managed module size

  6. Shared modules.

  Following these principles increases the likelihood of achieving acceptable levels of reliability and maintainability.

- **Structures flowcharts**, also called Nassi-Schneiderman diagrams, define modules in a system using the three basic process, decision and iteration structures. Each is assembled in a top-down fashion to specify the logic of a module or system.

- **Transaction analysis** is the technique of identifying transaction types of a system using them as units of design.

- **Transform analysis**, or transform-centered design, is the strategy for converting each piece of the data flow diagram into a structure chart. Transform analysis is a 'strategy' not an algorithm. Hence no fixed rules.

- **There are five steps involved in transform analysis :**

  1. **Drawing a data flow diagram**

  2. **Finding the central function of the data flow diagram**

  3. **First level factoring**

  4. **Factoring of Afferent, Efferent, and Transform branches**

  5. **Departures**

True men of action in our time, those who transform the world, are not politicians & statesman, but are scientists.

- W.H.Auden.

Chapter 6

Input / Output Design

At the end of this session, you will be able to:

- ➔ *Understand the objectives of input design*
- ➔ *Know data capture guidelines*
- ➔ *Design a source document*
- ➔ *Design a screen for input*
- ➔ *Understand the objectives of output design*
- ➔ *Know how to present information*
- ➔ *Design printed output*
- ➔ *Design screen output*

6.0 Introduction

Design of inputs and outputs are important features of design specifications. The input design is the link that ties the infc rmation system into the world of its users. Output, as you probably know, generally refers to the results that are generated by the system. For many end-users, Output is the main reason for developing the system and the basis on which they will evaluate the usefulness of the application.

In this session we will discuss objectives of input design and output design, Data capture guidelines, Presentation formats and methods for developing source document, printed outputs and screen outputs.

6.1 Input Design

The design decisions for handling input specify how data are accepted for computer processing. Analysts decide whether the data are entered directly, or by using source documents, such as variable forms where the data are transferred into the computer for processing.

6.1.1 Objectives Of Input Design

The Quality of system input determines the quality of systems output. Input specifications describe the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce results from accurate data, or they can result in the production or erroneous information. The input design also determines whether the user can interact efficiently with the system.

Six objectives guiding the design of the input focus on :

- Effectiveness
- Accuracy
- Ease to use
- Consistency
- Simplicity
- Attractiveness

All these objectives are important and can be attained by the use of basic design principles.

Effectiveness: This means that input forms and screens serve specific purposes.

Accuracy: refers to design that assures proper completion.

Ease to use: means that the forms and screen are straight forward and require no extra time to understand. This is especially required when automating manual systems.

Consistency: means that forms and screens should group data of similar nature together.

Simplicity : refers to keeping the forms and screen simple and uncluttered.

Attractiveness: input forms should be of appealing design, which should please the user.

6.1.2 Data Capture Guidelines

There are general guidelines that will assist the analyst in formulating an input design. The analyst should start by capturing only those items that must actually be input.

1. Variable data

Those data items that change for each transaction handled or decision made.

For example: The O.T hours of each employee varies, therefore it should be entered. On the other hand, the rate of O.T does not vary from employee to employee, so it need not be entered. The rate can be stored in the system and retrieved automatically when the O.T has to be calculated.

2. Identification data

The element of the data that uniquely identifies the record being processed. The identifying item in each transaction record is called a key.

For example, during entry of the variables, the employee must be identified, by the employee number. The employee name or even some other data element could be used instead, but it might not be unique and could thus cause error, or it may be difficult to enter.

*Besides knowing what to enter, knowing what should**not** be entered is equally important. Input procedures should not provide for the entry of the following data:*

1. Constant data:

This implies to data that are the same for every entry. For example, the number of working days for the month will be the same for all employees, it need not be input for every transaction. The analyst should instruct the programmer to write the software so that it directs the user to enter the no. of working days of the month before the process.

In cases where the date is required the clock/calendar in the computer can be used or the current date can be accepted at the start of the entry.

2. Details that the system can retrieve:

This refers to the stored data that are quickly retrievable from the system files.

For example, when entering the employees variable data, there is no need for the entry of employee's name , once the number is entered the name can easily and quickly be retrieved.

3. Details that the system can calculate:

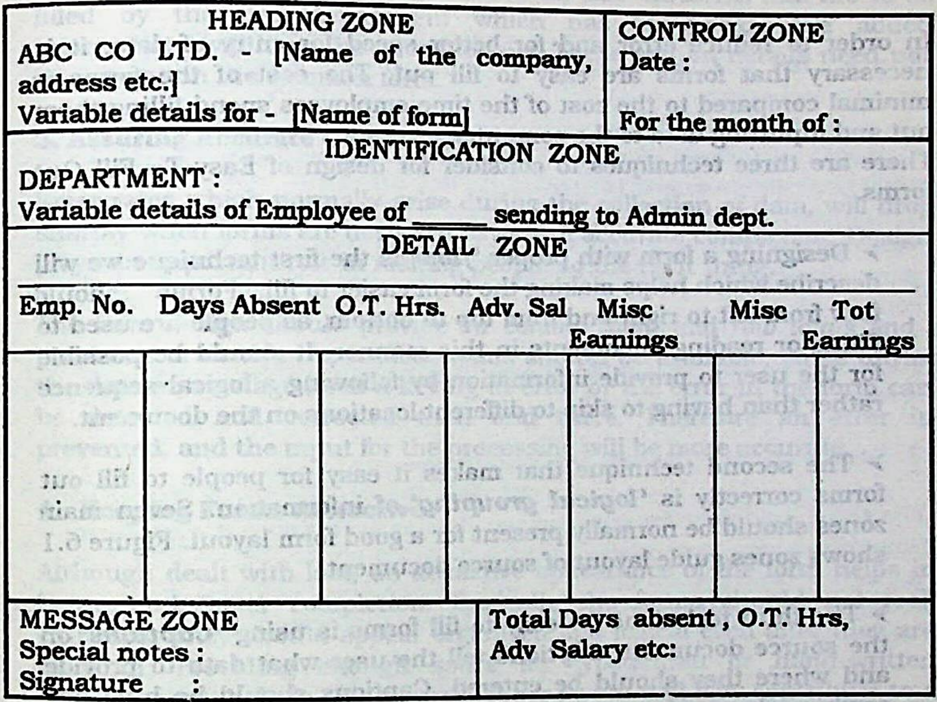
This refers to the results that can be produced by having the system use combinations of stored and entered data.

For example, to compute the O.T payable of an employee, the O.T hours of the employee has to be entered, the no. of working days can be accepted at the start, and the basic salary details can be retrieved from the employee master file, using the employee no., having this the O.T can be computed by applying the formula.

6.1.3 Design Of Source Document

This is also known as the input form. The source document is the form on which data are initially captured, i.e. recorded.

For example, the variable data input form shown in figure 6.1 is the source document for recording variable data that will enter the system. By, definition, they are pre-printed or duplicated papers that require people to fill in responses in a standard way.



HEADING ZONE
ABC CO LTD. - [Name of the company, address etc.]
Variable details for - [Name of form]

CONTROL ZONE
Date :
For the month of :

IDENTIFICATION ZONE
DEPARTMENT :
Variable details of Employee of ______ sending to Admin dept.

DETAIL ZONE

Emp. No.	Days Absent	O.T. Hrs.	Adv. Sal	Misc Earnings	Misc	Tot Earnings

MESSAGE ZONE
Special notes :
Signature :

Total Days absent , O.T. Hrs, Adv Salary etc:

Fig. 6.1

Zones to guide layout of source document

In order to design useful forms the following four guidelines for form design should be followed.

1. Make forms easy to fill out.

2. Ensure that forms meet the purpose for which they are designed.

3. Design forms to assure accurate completion.

4. Keep forms attractive.

There are a number of means to achieve each guideline for form design. Each of the four guidelines is considered separately below:

1. Make Forms Easy To Fill Out

In order to reduce error and for better speed for entry of data, it is necessary that forms are easy to fill out. The cost of the forms is minimal compared to the cost of the time employees spend filling them out and inputting data to the computer.
There are three techniques to consider for design of Easy To Fill Out forms.

> Designing a form with proper *'flow'* is the first technique we will describe which helps making the form easier to fill. Forms should flow from left to right and from top to bottom, as people are used to filling or reading documents in this manner. It should be possible for the user to provide information by following a logical sequence rather than having to skip to different locations on the document.

> The second technique that makes it easy for people to fill out forms correctly is *'logical grouping'* of information. Seven main zones should be normally present for a good form layout. Figure 6.1 shows zones guide layout of source document.

> The third technique for easy to fill forms is using *'Captions'* on the source document. Captions tell the user what data to provide and where they should be entered. Captions should be brief but easily understood, with standard terms that all persons using the form should know. Abbreviations should generally be avoided.

Inclusion of simple examples will help eliminate unnecessary questions about what information to provide.

For example, in asking for date of birth, the form might specify how the date should be provided such as "MONTH, DATE, YEAR" or "MM/DD/YY" or "MM/DD/19YY" . This will help the person concerned fill the date in the required format.

A well designed source document is easily completed and allows the process of actually recording the data to be rapid.

2. Meeting The Intended Purpose

Forms are intended to serve one or more purposes in recording, processing, storing and retrieving of information for the business. Sometimes it is required to provide different information to different departments or users while sharing some basic information. This is where special forms are useful. These special forms consist of the common basic information required by all, and the forms that are to be filled by the user/department which has to provide the added information, will have the added details. These add on details need not be present on the common form.

3. Assuring Accurate Completion

Error rates which normally arise during the collection of data, will drop sharply when forms are designed to assure accurate completion. Design plays an important role in making people do the right thing.

The form design should provide for *column totals* and *row totals* and a provision to *double check these totals* should be available, so as during the data entry stage itself a keying in error or the error in the form can be detected and corrected then and there. Therefore an error is prevented, and the input for the processing will be more accurate.

4. Keeping Forms Attractive

Although dealt with last, an attractive appearance of the form helps in better and faster completion. Basically the forms should not look cluttered. They should appear organised and logical even after they are filled in. Providing enough space for typewritten or hand-written responses will help in this regard. Proper layout and flow contribute to a form's attractiveness.

6.1.4 Design Of Screen Design

The completed source document or form is used to enter the data for the system. A good screen design, like a good form design, is an important instrument for steering the course of work.

Most of the principles discussed for good form design should be used for good screen design as well. The format of the screen should resemble the form as closely as possible.

Four guidelines for screen design are important:

1. Keep the screen simple.

The VDT screen should show only that data which is necessary for the particular action being undertaken.

2. Keep the screen presentation consistent.

A consistent screen design is necessary for a good screen design. If users are working from paper forms, screens should follow what is shown on paper.

Screens can be kept consistent by locating information in the same area each time a new screen is accessed.

Information that logically belong together should be consistently grouped together: Name and address go together, not name and pin code.

For example, the screen for railway booking should look similar to the reservation form being used.

3. Facilitate user movement among screens.

The third guideline for good screen design is to make it easy to move from one screen to another.

Example : Hot keys can be used for faster movement. Messages at the bottom of the screen.

4. Create an attractive screen.

This is the fourth guideline for good screen design. If users find screens appealing, they are more likely to be

- more productive
- need less supervision
- make less errors

Some of the design principles used for forms apply here as well. Screens should draw the user into them and hold their attention. Like good forms, the screens should not be cluttered, use of multiple screens is advisable rather than jamming all the data into one screen. Use logical flow in the plan of your screens.

6.2 Output Design

One of the most important features of an information system for users is the *output* it produces. Output is information delivered to users through the information system. Without quality output, the entire system may appear to be unnecessary that users will avoid using it. *Users generally merit the system solely by its output.*In order to create the most useful output possible, the systems analyst works closely with the user through an interactive process, until the result is considered to be satisfactory.

Therefore an effective output design is an important feature of design specification.

6.2.1 Objectives Of Output Design

Since useful output is essential to gaining use and acceptance of the system, the systems analyst should try and follow the following objectives which are useful for designing acceptable outputs.

1. Design output to serve the intended purpose.

2. Design output to fit the user.

3. Deliver the appropriate quantity of output.

4. Assure that output is where it is needed.

5. Provide output on time.

6. Choose the right output method.

1. Design Output to serve the intended purpose

All outputs should have a purpose. During the analysis phase the analyst finds out the purpose of the output. The output should be designed to meet those purposes. If the output is not functional, it should not be created, since there are costs of time and materials associated with all output from the system.

2. Design output to fit the user

The analyst by using the fact-finding techniques discussed earlier, should determine what information to present, which the users will need and prefer. The output should be what the user wanted.

The users of certain outputs could be external users. These have specific requirements governing content, format and media requirements that are unchangeable, for example. the monthly P.F statement, a copy of which will be required to be sent to the PF office in their standard specified format. No change to this format can be made.

3. Delivering The Appropriate Quantity Of Output

Part of the task of designing output is deciding what quantity of output is proper for users. The system analyst must provide what each person needs to complete his or her work. *No one is served if excess information is given.*

Incase there is a query request for an employee's yearly salary, then this could be displayed in the following manner ; rather than cluttering the screen with an entire year's salary details, each of the twelve screens might provide a month's salary details and the summary information available on separate screens.

It is necessary to keep the decision maker in mind when deciding about quantity of output. They often will not need great amount of output, especially if there is an easy way to access more. High and low quantities of data also suggest whether print or display should be used.

4. Making Sure The Output Is Where It Is Needed

Most commonly outputs are either printed on paper or displayed on screens. Outputs often have to be distributed to the users.

The increase in on-line, screen-displayed output, that is personally accessible, has cut down on the problem of distribution. Appropriate distribution is still an important objective for the systems analyst.

To be used and useful, output must be presented to the right user. No matter how well-designed the reports are, *if they are not seen by the person who requires it they have no value.*

5. Providing The Output On Time

One of the most common complaints of users is that they do not receive information in time to make necessary decisions.

Some reports are required on a daily basis, some monthly, others only annually and others as and when a request is made. Accurate timing of output can be critical to business operations.

6. Choosing The Right Output Method

Whether the output is a formatted report or a simple listing of contents of a file, a computer process will produce the output.

System output may be :

- A report
- A document
- A message

Depending on the circumstances and the contents, the output maybe displayed or printed. Output contents originate from these sources:

- Retrieval from a data store
- Transmission from a process or system activity
- Directly from an input source.

Choosing the right output method for each user and purpose is another objective in designing outputs.

Exercise - 1

1. What are the six objectives for input design .
2. ____________________ is an important feature of design specification.
3. What are the sources from which the output contents originate?

Ans: 1) Effectiveness, Accuracy, Ease to use, Consistency, Simplicity, Attractiveness
2) Effective output design

3) a) retrieval from a data store b) transmission from a process of system activity c) directly from an output source.

6.2.2 How To Present Information

How the information is presented will determine whether the output is clear and readable, the details convincing and the decision making fast and accurate. We will now discuss guidelines for presenting tabular and graphic information.

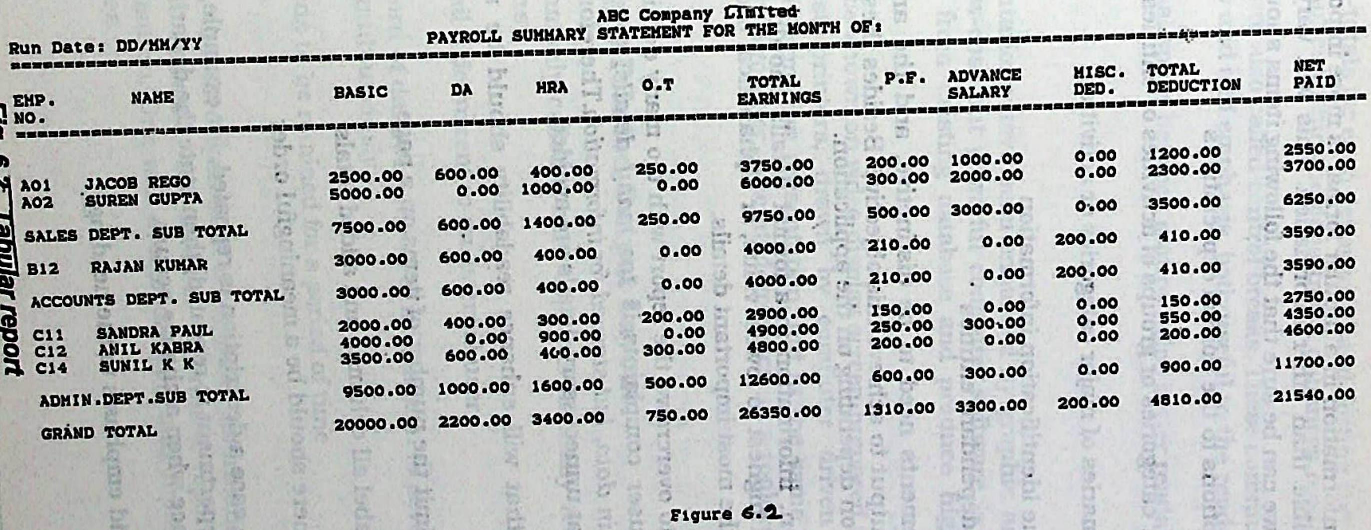
Tabular Format

Tabular format implies to a row-and-column format as shown in figure 6.2.

The word **'report'** normally suggests a tabular format to many people. In general; the tabular format should be used under the following conditions:

- Details dominate and few narrative comments or explanations are needed
- Details are presented in discrete categories
- Each category must be labelled

Totals must be drawn or comparisons made between components



```
                                     ABC Company Limited
                     PAYROLL SUMMARY STATEMENT FOR THE MONTH OF:
Run Date: DD/MM/YY
==========================================================================================================
EMP.      NAME              BASIC      DA       HRA      O.T     TOTAL      P.F.   ADVANCE   MISC.  TOTAL      NET
NO.                                                              EARNINGS          SALARY    DED.   DEDUCTION  PAID
==========================================================================================================

A01   JACOB REGO           2500.00   600.00   400.00   250.00   3750.00   200.00  1000.00   0.00  1200.00   2550.00
A02   SUREN GUPTA          5000.00     0.00  1000.00     0.00   6000.00   300.00  2000.00   0.00  2300.00   3700.00

SALES DEPT. SUB TOTAL      7500.00   600.00  1400.00   250.00   9750.00   500.00  3000.00   0.00  3500.00   6250.00

B12   RAJAN KUMAR          3000.00   600.00   400.00     0.00   4000.00   210.00     0.00 200.00   410.00   3590.00

ACCOUNTS DEPT. SUB TOTAL   3000.00   600.00   400.00     0.00   4000.00   210.00     0.00 200.00   410.00   3590.00

C11   SANDRA PAUL          2000.00   400.00   300.00   200.00   2900.00   150.00     0.00   0.00   150.00   2750.00
C12   ANIL KABRA           4000.00     0.00   900.00     0.00   4900.00   250.00   300.00   0.00   550.00   4350.00
C14   SUNIL K K            3500.00   600.00   400.00   300.00   4800.00   200.00     0.00   0.00   200.00   4600.00

ADMIN.DEPT.SUB TOTAL       9500.00  1000.00  1600.00   500.00  12600.00   600.00   300.00   0.00   900.00  11700.00

GRAND TOTAL               20000.00  2200.00  3400.00   750.00  26350.00  1310.00  3300.00 200.00  4810.00  21540.00


                                              Figure 6.2
```

Fig. 6.2 Tabular report

Certain information in a tabular format is more important and should be more visible than other information. This will vary by application. In general, we can be sure that the following items should stand out:

- Exceptions to the normal expectations
- Major categories or groups of activities or entities
- Summaries of major categories or activities
- Unique identification information
- Time-dependant entities

These elements need to be distinctive, and the analyst must design tabular output to achieve this objective. Besides these others could also be added on depending on the application.

In addition, information in a format details should be in a meaningful order (i.e. highest to lowest value, or alphabetical order), making it easy to locate the most important details.

It is easy to overcrowd the report with too many details. One of the most frequent user complaints is *too many details; too little information* or *drowning in data, but starved for information.*Therefore it is necessary to ensure that unnecessary details are avoided.

Features that will enhance readability should be used. These could include:

- ➢ Limit the number of items on a page
- ➢ Label all columns and added totals
- ➢ There should be a meaningful order
- ➢ In case a description is repeated, for example, in figure 6.2 the department is repeated, then state the department name only once when a change occurs.
- ➢ Add emphasis to the heading

- Adding sub-totals for each department and sorting the employee names within the department enhances readability

- Sub-totals are also called control breaks. These communicate information and add emphasis, both useful to the reader.

- An additional space between groups helps better readability.

Graphic Format

Management presentations have been enhanced by graphic and visual for a long time. Low-cost but powerful computer software is available that will use data from existing database and produce high quality charts and graphs.

Graphic forms can be shown on video display screen prepared in several colours on low-cost printers, drawn by computer driven plotters. Graphics enhance information presentation, but at the same time, few are willing to accept computer generated graphics as a replacement for traditional reports.

6.2.3 Designing A Printed Output Layout

Printed output commonly called a hard copy are normally required for the following reasons:

- need to mail a document to an external user

- print a record of data or a report of information to several persons simultaneously.

- a copy needs to be retained for a period of time

Whenever possible, the development of a computer information system should reduce, not increase, the number of printed reports moving through the organisation. One well designed report may replace several poorly designed ones.

→ Printed Reports

Printed reports vary in size.
Normally these standard sizes are used :

9 1/2 by 11 inches

11 by 14 7/8 inches

8 by 14 7/8 inches

These sizes are for continuos forms. When developing the printed output the size is required. The layout that is prepared should cove the required area.

→ Special Output Forms

These are special forms pre-printed by a stationer. The variety is seemingly endless, since any colour or ink can be used. Many options on positioning of headings and logotypes are also available.

The analyst lays out the content of the report on a pre-printed form in the same manner as the computer-prepared report. Pre-printed forms convey distinctive corporate images through use of corporate colours and design. These form are extremely expensive compared to computer-generated report forms. Payslips are normally printed on pre-printed forms.

→ Developing A Printed Output Layout

It is essential to have the right information and detail on a report as well as selecting the proper output medium. The design of printed output layout is the arrangement of elements on the output medium. The analyst has to first build a mock-up of the actual report or document to show how it will appear after the system is in operation. The layout should show the location and position of the following

- ***:All variable information:***

This includes information that can vary each time the report is printed such as :

→ Item details

- **Summaries and totals**
- **Control breaks**

- ***All constant information:***

This includes information that remains the same whenever the report is printed. To indicate constant information, the analyst writes it on the layout form, one character per space.
The following fall under constant information:

- **Headings**
- **Document names ant titles**
- **Corporate names and addresses**
- **Instructions**
- **Notes and comments**

As mentioned earlier, the layout is a blueprint. This will guide the construction of programs in the development process.

Each variable must be accounted for in a program instruction.
The layout forms tells us precisely:

- **Where the data will be printed**
- **How far away it is from other details**
- **Whether there is enough space to include all the essential details without cluttering the appearance of the form**

Designing A Printed Output Layout

It is necessary to determine the items that will be included in the report before beginning the design of the layout. The requirements analysis provides this information. The data dictionary contains the necessary descriptive information such as the data item type and its length. Report layouts can be formed manually using paper layout forms. Figure 6.2 shows a sample report layout chart.

1. Headings

All output produced from an information system should have a title. The procedure for designing headings is :

- Enter the report title and heading on the layout using the specific columns in which you wish the information to appear.
- Centre the title.
- The layout includes a page number and date.
- This should appear on all the pages.
- The page number provides a quick reference for the users who work with data found at various locations throughout the report.
- After the entry of the headings is over, look at the actual contents of the report, the salary data.
- As the amount of space of each element to be printed is available in the data dictionary, the space each of them will occupy can easily be determined.
- Now calculate the total space that will be used.
- 130 spaces are available for printing. Once you know the total to be used then space the elements out within the 130 columns by leaving equal spaces between each element to be printed.
- Once you have determined the amount of space available for each data element and the space to be left between each, then enter the column headings for each of the data elements that are to be printed.
- This will be printed as part of the heading on each page of the report.
- It is a good practice to use an underline, dash, or some suitable symbol to separate the co. .mn headings from the start of the data. We in figure 6.2 have used a dash for this purpose.
- Every column should have a heading that describes its contents.
- The names and words should be spelt out. Avoid abbreviations.

2. Data Details

- Enter the description of the data below the respective column headings.

- Use 'X' for alphabetic or Alphanumeric and '9' for numeric data.

- If decimal points, currency symbols or other special symbols will be used then mark them accordingly.

- For a description which is long, for example Employee name, you could either

  * fill up 30 'X's OR

  * mark the first and the last column with a 'X' and connect the two with a straight line. OR

  * mark it as "30 X's"

- Although reports may continue for pages, you define the detail line only once.

- The wavy line descending from each data item on the paper layout form indicates that it is repeated as many times as necessary on each page of the report.

3. Summaries

Some report designs specify summary information, column totals, or sub totals.

- These are marked in the same manner as just mentioned.

- The salary report contains a department wise summary for total earnings, total deductions and net pay, with a grand total for all departments.

- The principle for showing the summaries remain the same.

- Label all titles and headings as you wish them to appear.

- Denote variable data by 'X' or '9' depending on the field type.

- ✈ Indicate the maximum length of the field.

The steps mentioned above are used in the design of all outputs(screen as well).

The principles for indicating field length, subtotals and other data are unchanged.

4. Guidelines

There are many guidelines that will make the analyst's job easier and more important ensure the users of receiving an understandable report. We have used several which are summarised here:

- ☞ Reports and documents should be designed to read from left to right and from top to bottom.
- ☞ The most important items should be easiest to find. (employee number is the most important item in the salary report, since it identifies the employee. It is placed on the left margin.
- ☞ All pages should have a title and page number and show the data the output was prepared.
- ☞ All columns should be labelled.
- ☞ Abbreviations should be avoided.

6.2.4 Designing Screen Output

In section 6.1.4 we have discussed guidelines for designing input screens, the same guidelines also apply for designing screen outputs. The contents changes.
Screen outputs differ from printed outputs in the following ways:

- ✈ A VDT is not permanent.
- ✈ They can be more specifically targeted to users.
- ✈ It is available on a more flexible schedule
- ✈ It is not portable

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

- Screen outputs could be changed through direct interaction.
- Screens require you to give instructions to the user on how to use the display. The user needs to be instructed on
  - which keys to hit when they want to continue reading further screens
  - how to end the display
  - how to interact with the display if possible.
- With printed outputs, you can assume that people know
  - how to search through a report
  - how to turn to the next page
  - what steps to take when finished with the report.
- Access to screen displays may be controlled through the use of a password. Certain information could be kept hidden from certain users i.e. information not meant for a particu!ar user need not be shown to him. This could be identified by the password. Further, certain users(again depending on the password) could be allowed to interact while others can only view the output. Whereas, distribution of printed output is controlled by other means.

Screen Design

Visual Display Screens typically have 80 columns with 24 lines. We will assume this standard size for our examples. As mentioned earlier. the information should read from left to right and from top to bottom. The most important details should be easiest to find. Titles and headings should be consistently positioned.

In designing a screen we need areas for :

1. Headings and titles
2. Content of the display
3. Messages and instructions
4. Sometimes explanations for information in the report.

A status line provides the user with information about the program: For example, a sort routine is half finished, a report is printing and so on.

The information need not be detailed, as it is only a reference.

Information instructing the users how to proceed, is generally shown at the bottom of the screen.

The principles of designing a screen layout chart, remain the same as that of the print layout chart.

Exercise - 2

1. What are the things which are included in a report of printed layout.
2. Which are the details which come in variable information and which comes under constant information :-
Headings, separator, control breaks, notes, summary, totals, instructions, document names/titles

Ans: 1) Headings, data details, summaries and guidelines
2)

Variable information	**Constant information**
Summaries, Totals, Control breaks, Separator	**Headings, Document names/titles, Notes, Instructions**

Summary

This chapter has covered elements of input design for screens and source documents. Six objectives are to be followed for well designed inputs:

- effectiveness
- Accuracy
- Ease to use
- Consistency
- Simplicity
- Attractiveness

The analyst should start by capturing only those items that must actually be input. In order to design useful forms the following four guidelines for form design should be followed:

1. Make forms easy to fill out
2. Ensure that the forms meet purpose for which t .ey are designed
3. Design forms to assure accurate completion
4. Keep forms attractive.

Design of useful forms and screens overlaps in many ways, but there are some distinctions. Screens display a cursor which continually orients the user. The four guidelines for well designed screens are:

1. Keep screens simple
2. Keep the screen presentation consistent
3. Facilitate user movement among screens
4. Create an attractive screen.

Proper flow of both forms and screens is important.

Output is any useful information or data delivered by the information system to the user. The analyst has six main objectives in designing output. They are:

1. Design Output to serve the intended purpose.

2. Design output to fit the user.

3. Deliver the appropriate quantity of output.

4. Assure that output is where it is needed.

5. Provide output on time.

6. Choose the right output method.

The information could be presented in a tabular format or graphic format. The word 'report' normally suggests a tabular format to many people.

Chapter 7

Testing, Implementation & Maintenance

At the end of this session, you will be able to:

- *Know testing strategies, types of test data, and levels of testing*
- *Have an idea of verification and validation*
- *Know the importance of training*
- *Get an idea of the various types of conversion methods*
- *Know what is systems maintenance*
- *Know the details of documents that are output at different phases of the systems development life cycle*
- *Know the importance of structured walkthroughs and when they should be conducted*

7.0 Introduction

The techniques and guidelines introduced in the preceding sessions if properly followed, will produce low-failure systems. However, even if techniques are followed, the analyst must not assume that the necessary quality standards have been met. Quality Assurance is the review of software products and related documentation for completeness, correctness, reliability, and maintainability.

7.1 Managing Quality Assurance

Quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability.

It also includes assurance, that the system meets the specification and the requirements for its intended use and performance.

Quality assurance can be done by:

- Testing
- Verification And Validation.

7.1.1 Testing

Testing is generally done at two levels - testing of individual modules and testing of the entire system (systems testing).

During systems testing, the system is used experimentally to ensure that the software does not fail, i.e., that it will run according to its specifications and in the way users expect.

Special test data are input for processing, and the results examined. A limited number of users may be allowed to use the system so analysts can see whether they use it in unforeseen ways. It is preferable to discover any surprises before the organization implements the system and depends on it.

Testing is done throughout systems development at various stages (not just at the end). It is always a good practice to test the system at many different levels at various intervals, that is, sub-systems, program modules as work progresses and finally the system as a whole. If this is not done, then the poorly tested system can fail after installation.

As you may already have gathered, testing is a very tedious and time consuming job. For a test to be successful the tester, should try and make the program fail. The tester maybe an analyst, programmer, or specialist trained in software testing. One should try and find areas in which the program can fail. *Each test case is designed with the intent of finding errors in the way the system will process it.*

Thorough testing of programs do not guarantee the reliability of systems. It is done to assure that the system runs error free.

7.1.1.1 Testing Strategies

There are two general strategies for testing software. This section examines both the strategies of code testing and specification testing.

☞ Code Testing

The code testing strategy examines the logic of the program. For this testing method, the analyst develops test cases that result in executing every instruction in the program or module, that is, every path through the program is tested.

A *path* is a specific combination of conditions that is handled by the program. The testing of every path in the program is not always possible as there could be several thousands, and financial and time limitations will not permit this. Generally, all the frequently used paths undergo testing.

☞ Specification Testing

To perform specification testing, the analyst examines the program specifications wherein states what the program should do and how it should perform under various conditions. Then, test cases are developed for each condition or combination of conditions and submitted for processing.

By examining the results, the analyst can determine whether the program performs according to its specified requirements. This testing does not look into the program to study the code or path, *it looks at the program as a whole*. The assumption here is that, if the program meets the specifications it will not fail.

7.1.1.2 Types Of Test Data

There are two very different sources of test data:

- ➢ Live
- ➢ Artificial

Both have advantages and disadvantages.

Using live test data

Live test data are those that are actually extracted from organization files. This shows you how the system will perform on typical data. Although, the data may be the best, it is difficult to obtain sufficient amounts to conduct extensive testing. All combinations and conditions of the system are not tested with this data. This may not contain values that may cause system failure.

Using Artificial data

Artificial test data are solely for test purposes. They are to be generated to test all combinations of formats and values. They are generated using the utility programs of the information systems. Using this type of data all logic and control paths through the program can be tested.

For best results, the artificial test data should be generated by persons other than those who wrote the programs. Automated test data generators are also available.

7.1.1.3 Levels Of Testing

As already mentioned, testing is carried out at different levels and at various intervals.

☛ Unit Testing

This involves the tests carried out on modules/programs which make up a system. This is also called as program testing. The units in a system are the modules and routines that are assembled and integrated to perform a specific function.

In a large system, many modules at different levels are needed. Unit testing focuses first on the modules, independently of one another, to locate errors.

The programs should be tested for correctness of logic applied and should detect errors in coding. For example in the payroll system, all the calculations should be tested by feeding the system with all combinations of data.

Valid and invalid data should be created and the programs should be made to process this data to catch errors. For example in the payroll system, the employee no. consists of three digits, so during testing one should ensure that the programs do not accept anything other than a 3 digit code for the employee no..

Another e.g. for valid and invalid data check is that, in case a three digit no. is entered during the entry of transaction, and that number does not exist in the master file, or if the number entered is an exit case, then the program should not allow the entry of such cases.

All dates that are entered should be validated. No program should accept invalid dates. The checks that need to be incorporated are : in the month of Feb the date cannot be more than 29. For the months having 30 days one should not be allowed to enter 31.

All conditions present in the program should be tested. Before proceeding one must make sure that all the programs are working independently.

☞ Systems Testing

When unit tests are satisfactorily concluded, the system as a complete entity must be tested. At this stage, end-users and operators become actively involved in testing. While testing one should also test to find discrepancies between the system and its original objective, current specifications and systems documentation.

For example, one module may expect the data item for employee number to be numeric field, while other modules expect it to be a character data item. The system itself may not report this error, but the output may show unexpected results. A record maybe created and stored in one module, using the employee number as a numeric field. If this is later sought on retrieval with the expectation that it will be a character field, the field will not be recognized and the message REQUESTED RECORD NOT FOUND will be displayed.

Systems testing must also verify that file sizes are adequate and that indexes have been built properly. Sorting and reindexing procedures assumed to be present in lower-level modules must be tested at the systems level to see that they in fact exist and achieve the results modules expect.

7.1.1.4 Special Systems Tests

There are other tests that are in a special category as they do not focus on the normal running of the system. They are listed below:-

- ***Peak Load Test***

  This is used to determine whether the system will handle the volume of activities that occur when the system is at peak of its processing demand. For instance when all terminals are active at the same time.

  This test applies mainly for on-line systems. For example, in a banking system, analyst want to know what will happen if all tellers sign on at their terminals at the same time before start of the business day. Will the system handle them one at a time without incident, will it attempt to handle all of them at once and be so confused that it 'locks up' and must be restarted, or will terminal addresses be lost? The only way sure way to find out is to test for it.

- ***Storage Testing***

  This test is to be carried out tc determine the capacity of the system to store transaction data on a disk or in other files. Capacities here are measured in terms of the number of records that a disk will handle or a file can contain. If this test is not carried out then there are possibilities that during installation one may discover that, there is not enough storage capacity for transactions and master file records.

- ***Performance Time Testing***

  This test refers to the *response time* of the system being installed. Performance time testing is conducted prior to implementation to determine how long it takes to receive a response to a inquiry, make a backup copy of a file, or send a transmission and receive a response.

This also includes test runs to time indexing or resorting of large files of the size the system will have during a typical run or to prepare a report. *A system may run well with only a handful of test transactions, may be unacceptably slow when fully loaded.* This should be done using the entire volume of live data.

- ***Recovery Testing***

Analyst must never be too sure of anything. He must always be prepared for the worst. One should assume that the system will fail and data will be damaged or lost. Even though plans and procedures are written to cover these situations, they also must be tested.

- ***Procedure Testing***

Documentation and run manuals telling the user how to perform certain functions are tested quite easily by asking the user to follow them exactly through a series of events.

It is surprising how not including instructions about aspects such as, when to depress the enter key, removing the diskettes before putting off the power and so on, could cause problems. This type of testing brings out what is not mentioned in the documentation, and also the errors in them.

- ***Human Factors***

In case during processing, the screen goes blank, the operator may start to wonder as to what is happening can he could just about do anything such as press the enter key a number of times, or switch off the system and so on, but if a message is displayed saying that the processing is in progress and asking the operator to wait, then these types of problems can be avoided.

Thus, during this test we determine how users will use the system when processing data or preparing reports.

Exercise - 1

1. Quality Assurance is the review of software products and related documentation for ____________, ____________, __________, and ____________ .

2. Quality assurance can be done by:

3. Two types of test data are : ______________ and ____________.

4. When __________ are satisfactorily concluded, the _______ as a complete entity must be tested

5. There are other tests that are in a special category as they do not focus on the normal running of the system. List them.

Ans: 1) completeness, correctness, reliability, and maintainability.

2) Testing , Verification & Validation

3) Live data, Artificial data

4) Unit tests, system

5) Peak load test, Storage testing, Performance time testing, Recovery testing, Procedure testing, Human factors.

7.1.2 Verification And Validation

Verification testing runs the system in a *simulated environment* using *simulated data*. This simulated test is sometimes called *alpha testing*. The simulated test is primarily looking for errors and omissions regarding end users and design specifications that were specified in the earlier phases but not fulfilled during construction.

Validation refers to the process of using software in a *live environment* in order to find errors. The feedback from the validation phase generally produces changes in the software to deal with errors and failures that are uncovered. Then a set of user sites is selected that puts the system into use on a live basis. They are called beta test sites.

The *beta test* sites use the system in day-to-day activities. They *process live transactions* and *produce normal system output*. The *system is live in every sense* of the word, except that the users are aware they are using a system that can fail. But the transactions that are entered and the persons using the system are real. Validation may continue for several months. During the course of validating the system, failure may occur and the software will be changed. Continued use may produce additional failures and need for still more changes.

7.2 Implementation

After proper testing and validation, the question arises whether the system can be implemented or not. Implementation includes all those activities that take place to convert from old system to the new.

The new system may be totally new, replacing an existing manual or automated system, or it may be a major modification to an existing system. In either case, proper implementation is essential to provide a reliable system to meet organization requirements.

7.2.1 Training

A well designed system, if not operated and used properly could fail. Training the users is important, as if not done well it could prevent the successful implementation of an information system.

Throughout the systems development life cycle the user has been involved. By this stage the analyst should posses an accurate idea of the users that need to be trained. They must know what their roles will be, how they can use the system and what the system will do and will not do.

Both system operators and users need training. During their training, they need to be given a trouble shooting list that identifies possible problems and identifies remedies for the problem. They should be advised of the common mal functions that may arise and how to solve them.

The training should cover:

- ➢ familiarization with the processing system itself (that is the equipment used for data entry or processing
- ➢ training in using the application i.e. the software
- ➢ good documentation is essential, but this cannot replace training.

There is no substitute for hands on operation of the system while learning its use.

7.2.2 Conversion

Conversion is the process of changing from the old information system to the new or modified one. Conversions should be accomplished quickly as delays and long conversion periods cause frustration and the tasks of all involved including the analyst and user becomes more difficult.

There are four methods available for conversion. There is no single best way to proceed with conversion. Each method should be considered in light of the opportunities that it offers and problems that it may cause. Some situations dictate the use of one method over others, even though other methods maybe more beneficial. Each of the four conversion methods are discussed briefly below:

☞ Parallel Conversion

As the name implies, this refers to running the old system and the new system at the same time in parallel. This approach is most frequently used. This is the most secure method of converting from an old system to a new or modified one.

Both systems are run simultaneously for a specific period of time. When the new system is proven to be functioning as it should, then the old one is stopped. This method is best used when a computerized system replaces a manual one.

Advantages of this method are:-

- ➢ Offers greatest security. In the case of problems or errors in the new system, then the old system is there as backup.
- ➢ Users are more at ease as they do not have to make an abrupt change to the new system.

Disadvantages of this method are:-

- ➢ Doubles the operating costs
- ➢ Burdens employees involved with double workload
- ➢ In case the old system is not manual, then it is difficult to make comparisons with old and new outputs.

- Supposedly the new in an improvement on the old, therefore the outputs should differ.

- Employees faced with the choice between the two may opt for the old as they are more familiar with it.

☞ Direct Cutover

Direct Cutover means that on a specified date, the old system is dropped and the new system is put into use. The organization now relies fully on the new system. For direct cutover also known as direct changeover, to be successful, extensive testing is to be carried out beforehand. Direct cutover is best used in cases where some delays in processing can be tolerated.

Advantages of this method are:-

- Forces users to make the new system work

- There are immediate benefits from new methods and controls.

Disadvantages in this method are :-

- There is no other system to fall back on incase of serious problems or difficulties in the new system.

- Requires most careful planning.

- Users may resent being forced into using an unfamiliar system without recourse.

- There is no adequate way to compare new results with the old.

☞ Pilot System

In this method *a working version of the system is implemented in one part of the organization for example a department.* The users in this area typically know that they are piloting a new system and that changes can be made to improve the system.

Once the required changes are carried out and the system is complete, then it is implemented throughout the organization either all at once or phase by phase. Pilot approach is best used when new systems involve new techniques or drastic changes in the organization.

Advantages of this method are:-

- ➢ **Provides experience to the users and operators**
- ➢ **Provides live test data before implementation**

Disadvantages in this method are :-

Incase implementation is not handled properly then users may develop the impression that the system is not error free and may think it unreliable.

Phase-In-Method

The Phase-in-method is used when it is not possible to install a new system throughout the organization *all at once. Only one phase of the system is implemented at a time.* The file conversions, training of personnel, or arrival of equipment may not take place all at once. This may force the staging of implementation over a period of time. This could be weeks or months. Some users may start taking advantage of the new system earlier.

Advantages of this method are:-

- ➢ **Allows some users to take advantage of the system early**
- ➢ **Allow training and installation without unnecessary use of resources.**

Disadvantages in this method are :-

- ➢ **A long phase-in causes user problems whether the project goes well or not.**

7.2.3 Conversion Plan

This plan should be formulated in consultation with the users. The conversion plan includes a description of all activities that must occur to implement the new system and put it into operation. This includes identification of persons responsible and timetable for each activity that is to be carried out.

During the planning of conversion, the analyst should form a list containing all tasks, including the following :-

1. List all files for conversion.
2. Identify all data required to build new files during conversion.
3. List all new documents and procedures that go into use during conversion.
4. Identify all controls to be used during conversion. Establish procedures for cross-checking the old and the new systems.
5. Determine how team members will know if something has not been completed properly.
6. Assign responsibility for each activity.
7. Verify conversion schedules.

The conversion plan should anticipate possible problems and ways to deal with them.

7.3 Systems Maintenance

A system should be created whose design is comprehensive and farsighted enough to serve *current and projected* user needs for several years to come. Part of the analyst's expertise should be in projecting what those needs might be, and then building flexibility and adaptability into the system. *The better the system design, the easier it will be to maintain*and the maintenance cost will be low. Reducing the maintenance costs is a major concern, since software maintenance can prove to be very expensive. It is important to detect software design errors early on, as it is less costly than if errors remain unnoticed until maintenance is necessary.

Maintenance is performed most often to improve the existing software rather than to respond to a crisis or system failure. *As user requirements change, software and documentation should be changed as part of the maintenance work.* Maintenance is also *done to update software in response to the change made in an organization.* This work is not as substantial as enhancing the software, but it must be done. The system could fail if the system is not properly maintained.

7.4 Documentation

Documentation or Procedure Manuals explains how the system is designed and operates. Access to procedure manuals is necessary for new people learning the system, as well as a reminder to those who use the program infrequently. They may contain background comments, steps to accomplish different processes, instructions on how to recover from problems, and what to do next if something isn't working (trouble shooting).

To be useful, manuals must be up to date. A good manual will be used repeatedly as a reference. As such, it needs to be organized in a logical way with careful thought given to the circumstances that would call forth use of the manual.

In general a good manual should be:

- well organized
- should be easy to locate needed information
- all cases should be included
- manuals should be written in plain English.

Besides the manual's organization and clarity, careful thought should be given to the kinds of people who will be using the manual.

Documentation is to be done at various stages of the SDLC.

1. Feasibility Report

This report is the output of the feasibility study which is carried out at the onset of the system. It tells us that the system requested is feasible or not. The major purposes of this report are :

1. Identify the responsible users and develop an initial "scope" of the system.

2. Identify current deficiencies in the user's environment

3. Establish goals and objectives for the new system.

4. Determine whether it is feasible to automate the system and, if so, suggest some acceptable scenarios.

This report contains a brief description of the current system and an outline of the proposed system.

2. Functional Specifications

This is the *output of the requirements analysis phase* of systems development life cycle. The feasibility study forms the basis of this report. This document contains the following details :

Describes *'what'* system should do

It contains :

- ➢ Detailed DFDs: All levels of Physical and logical DFDs
- ➢ Data dictionary: detailing the data in the system
- ➢ Input formats

- copy of the input documents and screen output
- specifications such as general format of all the screens
- Source of Data - which specifies type of input form needed by the system

- **Available From - Gives sources from where inputs are to be collected**

- **Output specifications**

- **List of reports required and their details such as :**
- **Report name and object of report**
- **Frequency - how often it is used**
- **Period - period for which information is required**
- **Due Date - when the report is required**
- **Prescribed arrangement- Specification of the order in which information is required**
- **Unit - Specifies units of columns of the report**
- **Distribution - Distribution of the report to various department persons.**

- **Process specification - Decision trees, structured English, and decision tables that are used to describe the logic used in the processes.**

3. System Design Note

This should give the details of the design of the system. The topics that should be covered in this document are :

- **Scope of the system**
  - **systems limitations, systems objectives, major functions and constraints.**
- **Structure charts**
- **Program specifications - a short description of what the program does, what are inputs to the program, outputs of the program, calculations if any, program code listings, the program codes could contain comments to explain complex sections of the code.**

➢ Input layouts

➢ Output layouts

➢ Data dictionary

4. User Manual

This is a very important document. If the user/operator does not use the system in the proper manner, then the system could fail. This should detail out the procedural steps to be followed right from the start to the finish. It should also tell the users the difficulties that could crop up and how to overcome them.

The back-up procedures should also be included so as no valuable data is lost. All screens should be included and should contain details of what is to be entered at each stage.

The user manual should generally contain the following:

➢ Introduction

- what the system does
- system functions
- users of the system
  - system developers
  - system and configuration required
- limitations
  - size limitations
  - assumptions

➢ Principles And Procedures

- General
  - Outputs

- Inputs

- General procedures

  - Start up/ sign on

  - Backup

  - Shutdown

  - Formatting disks

➢ Tutorial - step-by-step walkthrough of example functions to be illustrated in the example

➢ Reference

- For each function

  - Function description

  - How and when used

  - Structure of command or screen

  - What happens

  - How to use

  - Errors and what to do

  - Examples

➢ For each error

- How recognized

- Meaning

- What to do about it

➢ Appendix
How to install the system before first use.

7.5 Structured Walkthrough / Formal Technical Reviews

A structured walkthrough is a planned review of a system or its software by persons involved in the development effort. The purpose of walkthroughs is to find areas where improvement can be made in the system or the development process. A walkthrough should be viewed by the programmers and analysts as an opportunity to receive assistance. As users and developers are involved in walkthroughs, the concept recognizes that systems development is a team process.

The structured walkthrough should be used throughout the systems development process as a constructive and cost-effective management tool, after the detailed investigation(analysis review) following design (design review), and during program development (code review and testing review).

All walkthroughs includes documentation that participants read and study prior to the actual walkthrough.

- **Analysis Review**

This is conducted to examine the functional specifications of the system, which is prepared after the analysis phase of the SDLC. This walkthrough is aimed at examining the func.ions, activities, and pr.cesses that the new system will handle. It emphasizes the inormation and processing requirements the proposed design should handle. If there are inconsistencies among the requirements stated by the users and those the analyst is proposing to meet through the new system or if there are vague specifications, the walkthrough should uncover them so they can be dealt with.

- **Design Review**

Design reviews, as the name suggests, focus on design specification for meeting previously identified systems requirements. The information supplied about the design prior to the session can be communicated using structured charts, N-S flowcharts, screen designs, input formats, output formats, document layouts.

The purpose of this walkthrough is to determine whether the proposed design will meet the requirements effectively and efficiently. If the participants find discrepancies between the design and requirements, they will point out and discuss them.

- **Code Review**

A code review is a structured walkthrough conducted to examine the program code developed in a system, along with its documentation. It is used for new systems and systems under maintenance. This does not deal with an entire software system, but rather with individual modules or major components in a program.

- **Post-implementation Review**

After the system is implemented and conversion is complete, a review of the system is usually conducted by users and analysts alike. Not only is this a normal practice, but it should be a formal process to determine how well the system is working, how it has been accepted, and whether adjustments are needed.

The review is also important to gather information for the maintenance of the system. Since no system is really ever complete, it will be maintained as changes are required because of internal developments, such as new legal requirements, industry standards, or competition. The post-implementation review provides the first source of information for maintenance requirements.

Exercise - 2

1. In Pilot system the ________ is tested in ________ of the organisation whereas in a Phase-in Method ________ of the ________ is tested in the organisation.
2. A ____________ is a planned review of a system of its software by persons involved in the development effort.
3. ____________ is the output of the requirements analysis phase of systems development life cycle.
4. ______________ and ________ are to be done at various stages of the SDLC.
5. ________ is the process of changing from an old system to a new one.

Ans : 1) System, one part. / a part , system
2) Structured walkthrough
3) Functional specification
4) Documentation and Testing
5) Conversion

Summary

The quality of an information system depends on its design, development, testing and implementation.

Quality assurance includes testing to ensure that the system performs properly and meets its requirements.

Special cases of **testing** are validation and verification. The purpose of testing is to find errors, not to prove correctness. Test cases, using live or artificial data, are processed by the software, and errors are reported.

Six special tests are :

- peak load test
- storage test
- performance time test
- recovery test
- procedure test
- human factors test

Each focuses on finding operation flaws in the system to prevent failure. Both *live* and *artificial data* are used to test the system.

Implementing a system, whether a new or modified one, consists of three primary activities of

- training
- conversion
- reviews.

Both users and operators need to be trained well for the system to function as it should.

Conversion is the process of changing from an old system to a new one. It must be carefully planned and executed.

Four methods are common:

1. Parallel systems

2. Direct cutover

3. Pilot approach

4. Phase-in

Maintenance is performed to update software in response to the changing organization.

Procedure manuals explains how the system is designed and operates. After the system is implemented and conversion is complete, a review of the system is usually conducted by users and analysts alike.

A **structured walkthrough** is a planned review of a system oı its software by persons involved in the development effort. The purpose of walkthroughs is to find areas where improvement can be made in the system or the development process.

Appendix - A

Case Study - Payroll System

For our case study, we have chosen the payroll system of ABC Co. Ltd., which is a very familiar and simple system. Reference to this case will be made through out the book.

☞ Organizational Overview Of ABC Co. Ltd.

"Payroll System" for ABC Co. Ltd.

ABC Co. Ltd. is the only sales outlet in Bombay for certain consumer goods. It basically consists of three departments.

1. **Sales department: Employees of this department are involved with the order processing system of the company. They carry out all the sales activities.**

2. **Accounts department : This department is involved with all the financial accounting aspects of the company.**

3. **Administration department: The major activity of this department, is payroll processing of all departments, and recruitment of new personnel.**

One of the jobs of admin department, is to calculate the payroll of the entire company. This so far is being done manually. The admin manager finds it very time consuming and feels that this system should be automated.

☞ Current manual system

At present the admin. department maintains *three separate employee registers*, for each of the three departments of the company. The *payroll is processed separately for each of the three departments.*

As and when any new employee joins, the *appointment form* containing all the employees standard details are sent to the admin. dept.. These details are entered in the respective department's register.

A few months ago, a particular new employee - Anil's, details from the appointment form were entered by the admin. clerk Joe, in the sales department register instead of the accounts department register. This mistake was realised after the entire processing was complete. When the accounts department could not locate Anil's payslip, they approached the admin. department. Joe, who had actually entered the register was absent. Another clerk Kumar, denied receiving Anil's appointment form. He checked the accounts department's employee register, and argued that if it had come, then the details would have been written into this register. The details were finally entered in the accounts department employee register, and the payslip for Anil was prepared.

After a week, when Joe resumed duty, he was informed by Kumar as to how he had to oblige the accounts department by preparing Anil's payslip after all the work was done. At that time Joe realised his mistake. He then had to cancel the information from the sales department employee register. Anil's payslip which had actually been prepared and sent to the sales department was found lying on the desk of the sales clerk, who distributed the payslips.

Just as the appointment form is sent to the admin, any changes in the employee details are also sent by the respective departments, and these changes are incorporated in the respective registers.

By the 20th. of every month, the departments send in their attendance registers and O.T registers. The accounts department in addition, sends in the advance salary voucher details of *all* employees.

Using this information Joe and Kumar work out No. of days absent, total OT hours for the month and total advance salary taken by each employee.

There have been many instances when these figures have been wrongly calculated, resulting in under or over payment. Naturally, the under payments have been reported while only few over paid cases have been reported.

Having completed this, the payroll of each department is calculated separately using the employee register details, days absent details, total OT hours and advance salary details, the monthly payroll statement for each department is prepared. This statement is used as basis for preparing payslips, bank statement and salary summary statements.

Again there are a number of instances when:

1. The figures appearing in the payslips do not tally with the payroll statement or the bank statement.

2. The figures appearing in the bank statement are wrong.

3. The salary summary statement does not tally.

All this causes a great amount of chaos and confusion every month. All these problems were reported to the management on various occasions. It was finally decided to computerise the Payroll Monitoring function of the company.

'Success Consultants' were entrusted with the development of the entire Payroll system of the company. They were asked first to carry out a feasibility study and hand over a feasibility report to the management.

Guidelines For Preparing A Feasibility Report For The Automation Of The Payroll System For ABC Co. Ltd.

☞ Identify the Responsible Users and Develop an Initial 'Scope' of the system

The analyst must identify two specific groups of end-users:

a. Those who use the system. In this case the officers and clerks who actually collect the data and calculate the payroll.

b. Those affected by the inputs and outputs of the system in study. In this case the Accounts department who should receive the accounts statement, and all those in the admin department who are involved with the inputs and outputs.

To develop the initial scope of the system you need to get a broad idea of the system in study. In this case we can identify the main process as 'the payroll monitoring process', which gets its inputs from the three departments of the company, which are the sales department, accounts department and the admin department. The outputs of the systems are sent back to the respective departments.

Besides, we need to develop an initial context diagram -which is a simple data flow diagram in which the entire system is represented by a single process. (The context diagram is described in chapter 2).

☞ Identify current deficiencies in the user's environment

As the payroll processing is being done separately for each of the departments, there is duplication of registers, and processes.

As there are separate registers maintained, very often the entries are entered in the wrong registers, causing duplication of work and confusion.

Errors in calculation result in, employees being wrongly paid, which need to be rectified in the following month.

Payslips and all the required reports have to be typed out. This is not only very laborious and time consuming but there are a number of errors found. Very often the statements do not tally.

☞ Determine Objectives for the new system

Here we will briefly list the functions of the new system.

➢ Maintenance of a employee details in a central location.
  Entry of new employee details.
  Updation of old employee details.

➢ Maintenance of a transaction details, and calculation of the days absent and Total OT hours of the month using the attendance details and daywise OT details obtained from the departments.

➢ Maintenance / check list printing o f current month's transaction details.

➢ Calculation of payroll and generation of the payslips.

➢ Generation of monthly reports :
  - Payslips
  - Cheques
  - Bank Statement (Payment advice)
  - Salary Summary Statement
  - Accounts Statement (for accounts)
  - PF Statement

➢ Generation of Yearly Reports:
  - Consolidated salary statement
  - Bonus Statement

- **Adhoc reports (as and when required)**
  - **- List of employee of particular grades**
  - **- List of employees whose basic salary is.between the given range.**

Determine whether it is feasible to automate the system

The Analyst discusses goals and objectives for the new system in a review with the admin. manager, officers, clerks handling the payroll and the company manager.

The admin manager considers the following three major areas to determine the feasibility of this project.

Technical feasibility **: He determines whether the current level of technology can support the proposed system. ABC Co ltd., already has hardware installed. This, at present is being used by the accounts department. They have agreed to spare one terminal to the admin department during their processing time. The current set up is sufficient for the processing of the payroll once a month and even for the adhoc reports as well as annual reports.**

Economic feasibility**: He measures the cost effectiveness of the project. He now need not invest in the hardware as it is already available. He will still need to consider the time spent by the systems analysis' team, the cost of doing a full systems study, cost of employee's time involved in the study, cost of the development of the software which has been entrusted to 'Success Consultants'.**

Operational feasibility **: He considers the extent the proposed system will fulfil his department's requirements. That is whether the proposed system covers all aspects of the working system and whether it has considerable improvements. In this case the employees of the admin department themselves have made the request to computerise the payroll system. They are very keen to see it operational.**

On having decided that they should go ahead, they request 'Success Consultants' tc carry out the Analysis and Design of the company's payroll system.

Data Flow Diagrams

Physical Context Diagram For Payroll Monitoring System

The DFD showing the general i.e. top layer of the system is called the 'Context diagram' .

CONTEXT DIAGRAM FOR PHYSICAL PROCESSING OF SALES DEPT

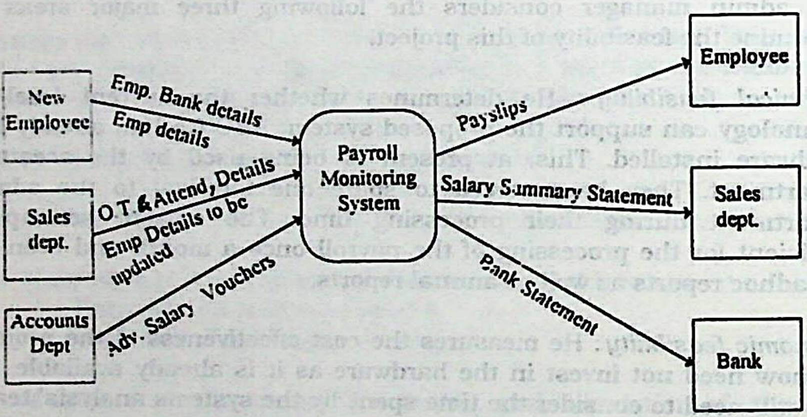


Fig 2.1

Figure 2.1 shows the Physical Context Diagram which describes the payroll monitoring system at a very general (top) level.

- The context diagram consists of only one process, data flows and entities.
- The single process in figure 2.1 is 'Payroll Monitoring System'.
- The context diagram defines the system in study i.e. it determines the boundaries.
- Each arrow, representing data flow, is labeled to show what data are being used.

- **New employee gives the new employee details and the employee's bank details to the system.**
- **The respective departments feed the system with the O.T details, attendance details and employee details which are to be updated.**
- **The accounts department give the advance salary details.**
- **The accounts department receives the accounts statement from the system.**
- **The salary statement is received by the respective department.**
- **The bank receives the bank statement**
  - **The employees receives the payslips.**
  - **When data moves to a process (i.e. data serves as input to the process) the arrow points towards the process to reflect the input.**
- **When the process produces data, the arrow points away from the process reflecting the output.**

Developing the First Level Physical Data Flo·7 Diagram For Our Payroll Monitoring System

The description of the payroll monitoring in the context diagram requires more details. We, now have to describe the system as we understand it at level 1.

Figure 2.2 shows the First Level Physical Data Flow Diagram of the payroll calculation of the Sales department of ABC Co. Ltd.

The same is applicable for the other two departments of the company. (only the department name will vary). As can be seen, this level of the DFD contains five processes.

FIRST LEVEL PHYSICAL DFD FOR PAYROLL CALCULATION OF SALES DEPT

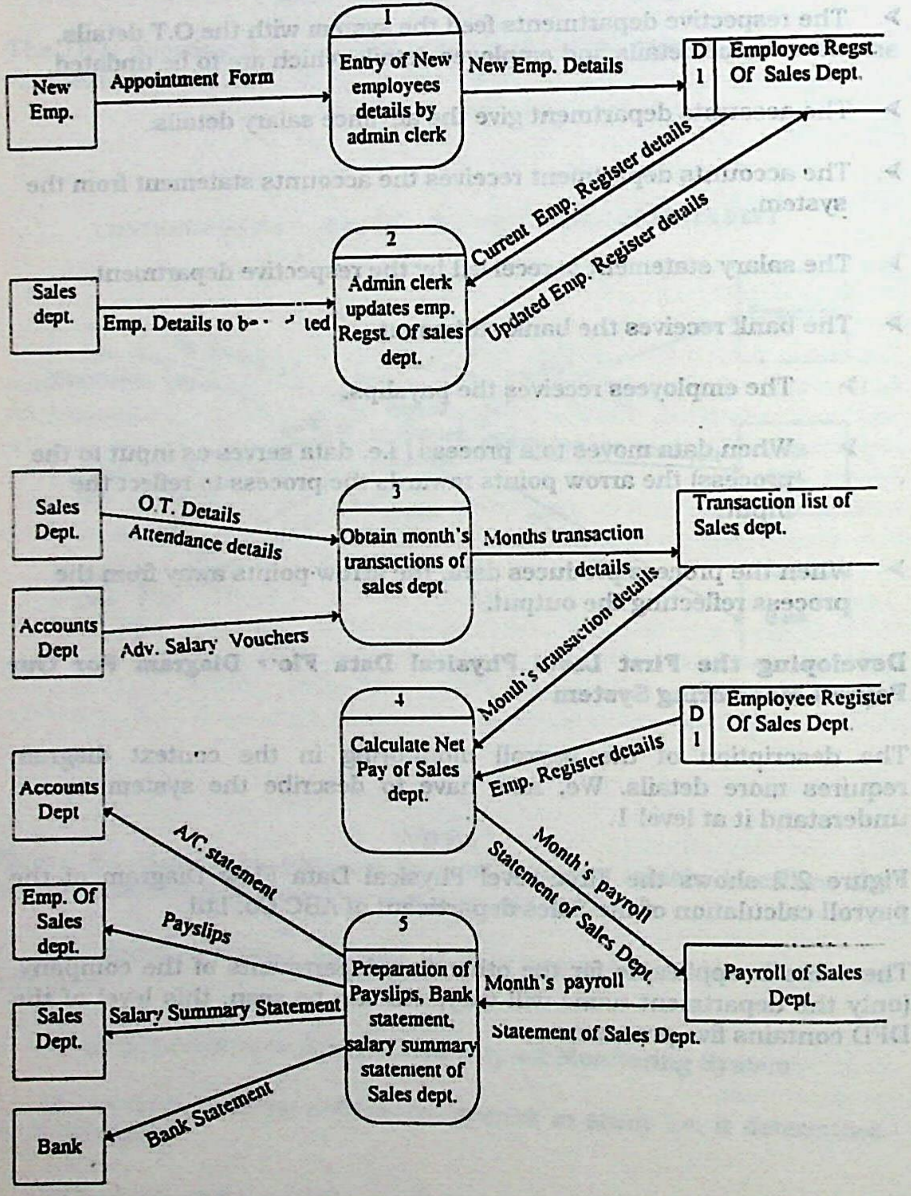

New Emp.
Appointment Form
1
Entry of New employees details by admin clerk
New Emp. Details
D 1
Employee Regst Of Sales Dept.
Current Emp. Register details
Updated Emp. Register details
Sales dept.
Emp. Details to be updated
2
Admin clerk updates emp. Regst. Of sales dept.
Sales Dept.
O.T. Details
Attendance details
3
Obtain month's transactions of sales dept
Months transaction details
Transaction list of Sales dept.
Accounts Dept
Adv. Salary Vouchers
Month's transaction details
4
Calculate Net Pay of Sales dept.
D 1
Employee Register Of Sales Dept
Emp. Register details
Accounts Dept
A/C statement
Month's payroll
Statement of Sales Dept
Emp. Of Sales dept.
Payslips
5
Preparation of Payslips, Bank statement, salary summary statement of Sales dept.
Month's payroll
Statement of Sales Dept.
Payroll of Sales Dept.
Sales Dept.
Salary Summary Statement
Bank
Bank Statement

Fig. 2.2

The following is seen in the DFD.

- **The Employee Data from the filled appointment form obtained from the new employee is entered into the respective department's employee register by the admin clerk.**

- **Admin clerk makes changes in the employee register wherever required.**

- **After obtaining the month's attendance details from the attendance register, day-wise O.T data from the O.T register and the advance salary form the advance salary, vouchers which is got form accounts department, the month's transaction list is written out.**

- **Now using this transaction list and the details from the employee register the payroll of each employee is calculated and stored in a payroll statement of the month.**

- **The payroll statement forms the basis for typing out the payslips and the bank statement and salary statement.**

- **The payslips are handed over to the employees of the respective departments and the bank statement is sent to the banks.**

Second Level Data Flow Diagram

We have just seen and discussed the First Level Physical DFD. From that we learn that the processes need to be *exploded* in the sense, need for more detail is required to get a clearer idea of the processes. We need to draw lower levels of the DFD to obtain this. Going from higher level to lower level is called 'exploding' a data flow diagram.

SECOND LEVEL DFD FOR OBTAINING MONTH'S TRANSACTIONS OF SALES DEPT.

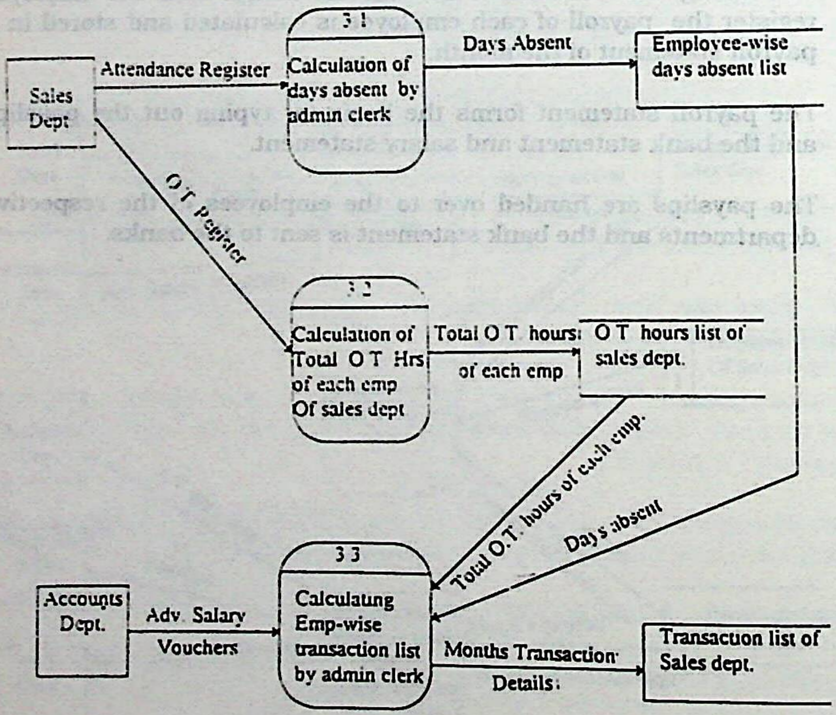
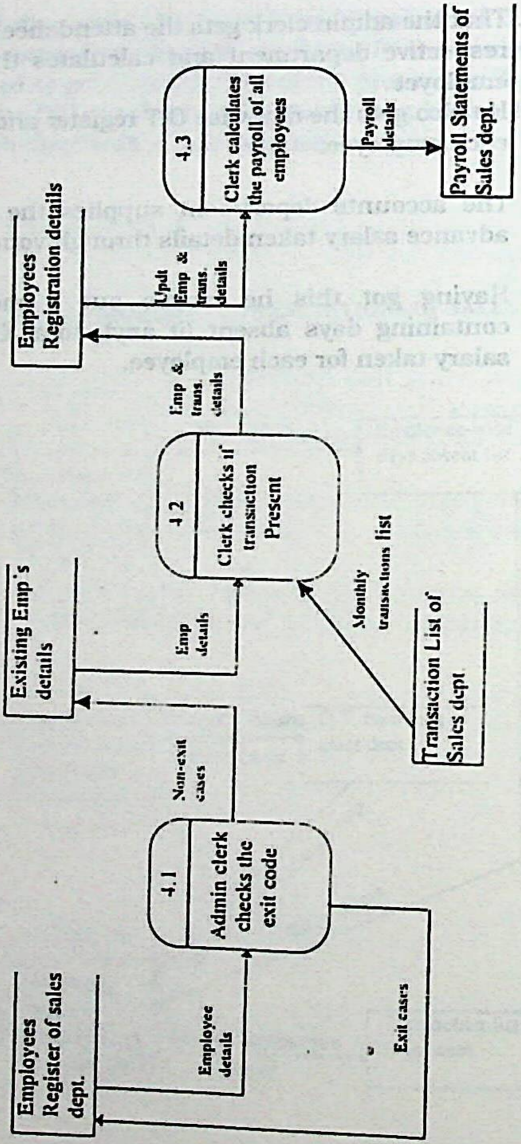

3.1
Sales Dept
Attendance Register
Calculation of days absent by admin clerk
Days Absent
Employee-wise days absent list
O.T. Register
3.2
Calculation of Total O T Hrs of each emp Of sales dept
Total O T hours of each emp
O T hours list of sales dept.
Total O.T. hours of each emp.
Days absent
3.3
Accounts Dept.
Adv. Salary Vouchers
Calculating Emp-wise transaction list by admin clerk
Months Transaction Details
Transaction list of Sales dept.

Fig. 2.3

Fig 2.3 shows the Second Level DFD for 'obtaining month's transaction' process for payroll calculation. From this we can gather :-

- **That the admin clerk gets the attendance register from the respective department and calculates the days absent of each employee.**
- **He also gets the day-wise O.T register and totals the O.T hours of each employee.**
- **The accounts department supplies the admin. clerk with the advance salary taken details through vouchers.**
- **Having got this he makes out a month's transaction list containing days absent (if any), total O.T hours and advance salary taken for each employee.**


Employees Register of sales dept.
Employee details
4.1
Admin clerk checks the exit code
Exit cases
Non-exit cases
Existing Emp's details
Emp details
4.2
Clerk checks if transaction Present
Monthly transactions list
Transaction List of Sales dept
Emp & trans. details
Employees Registration details
Updt Emp & trans. details
4.3
Clerk calculates the payroll of all employees
Payroll details
Payroll Statements of Sales dept.

Second Level DFD for 'Calculation' process of the payroll system

Fig. 2.4

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

Figure 2.4 shows the Second Level DFD for 'Calculation' process of the payroll system. From this we gather:-

- **That the admin clerk checks the employee register and by-passes all exit cases.**
- **For each non-exit case he then checks in the transaction list, if there is any transaction for that particular employee.**
- **He then uses the data present in the employee register and the transaction list (if data for the employee present) to calculate the payroll for the month. The month's payable data is thus obtained and stored in the payroll statement.**

Figure 2.5 shows the logical Context Diagram For Payroll Calculation of the full company.

LOGICAL CONTEXT DIAGRAM FOR PAYROLL PROCESSING

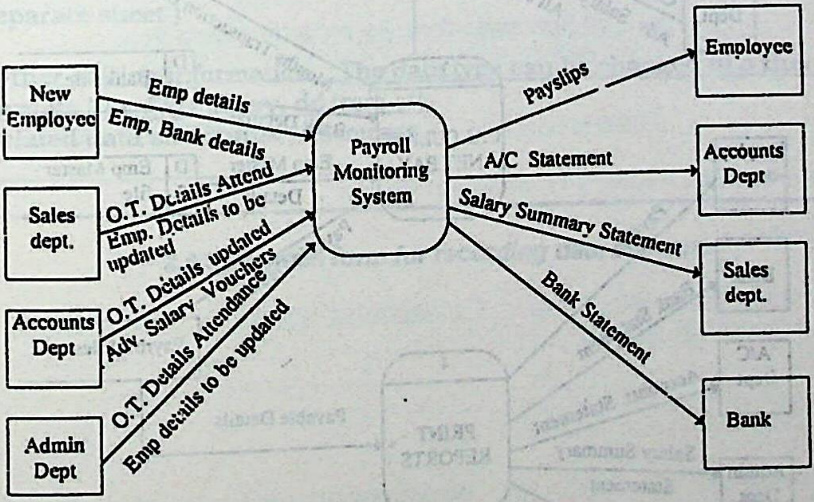


Fig 2.5

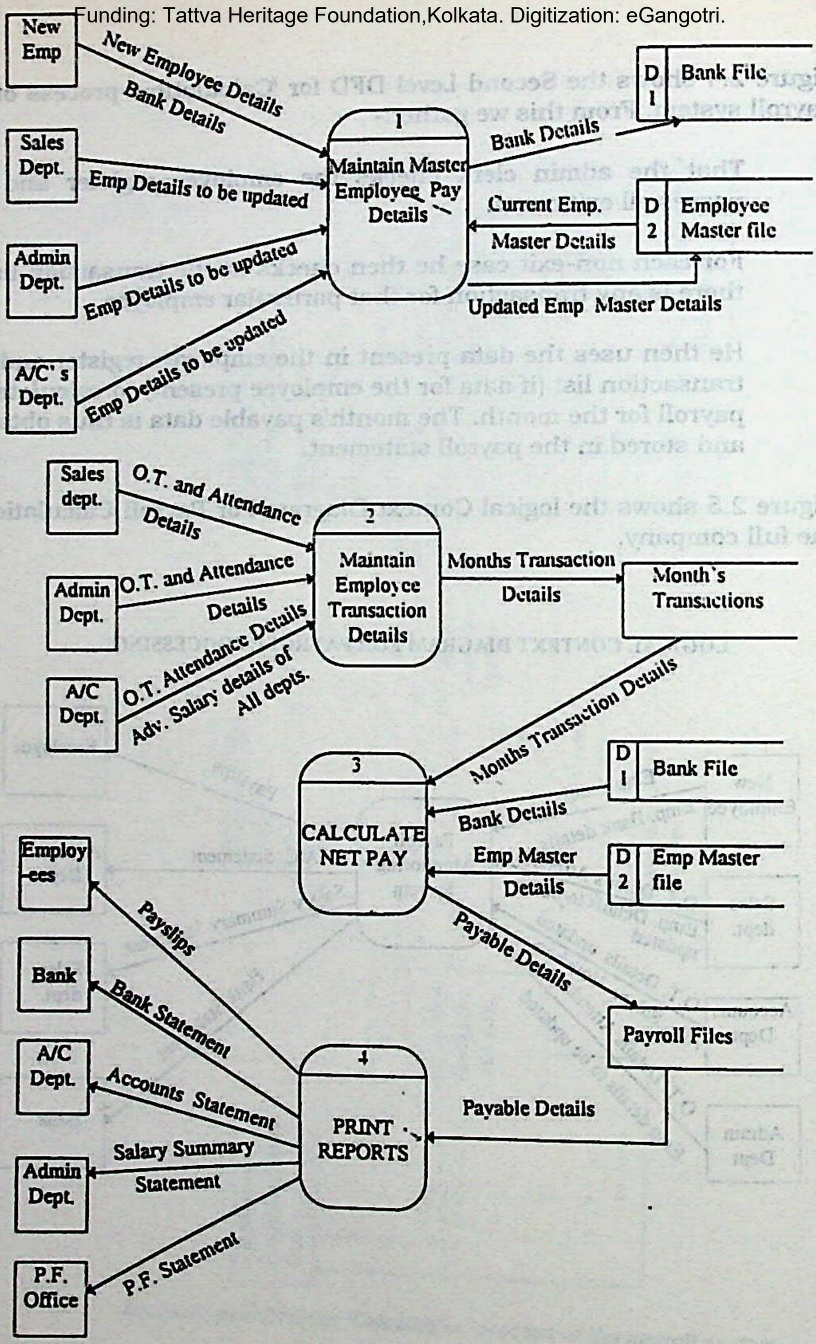
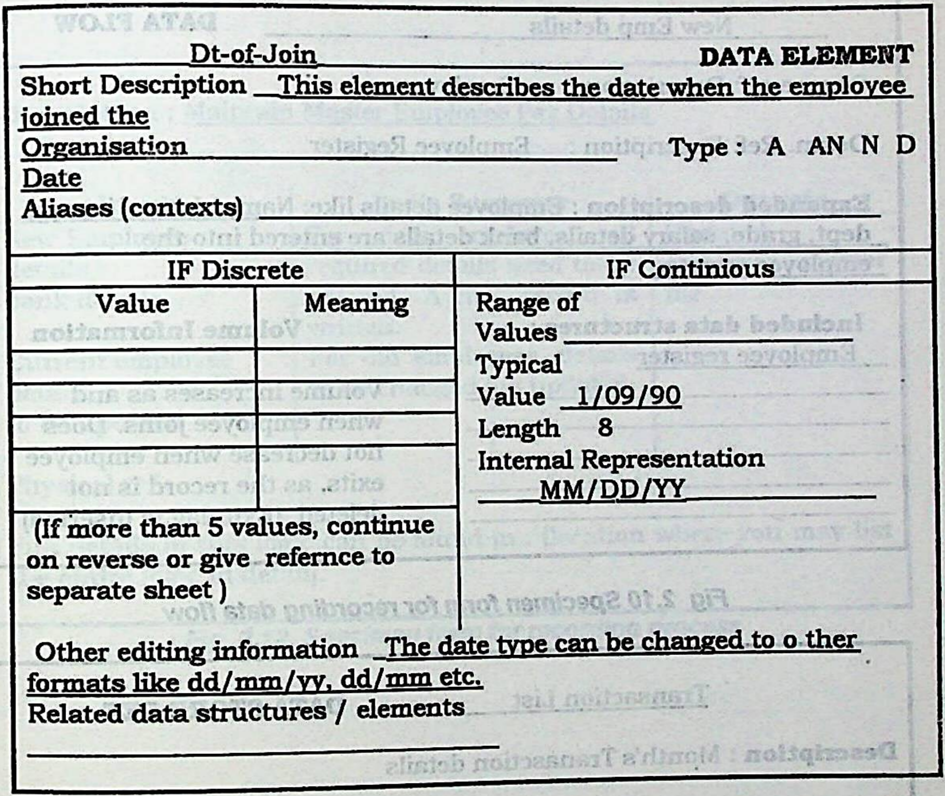


Fig 2.6 The logical DFD of the first level DFD of our payroll system

Data Dictionary



Dt-of-Join **DATA ELEMENT**

Short Description This element describes the date when the employee joined the Organisation Date

Type : A AN N D

Aliases (contexts) ______

IF Discrete		IF Continious
Value	Meaning	Range of Values ______ Typical Value 1/09/90 Length 8 Internal Representation MM/DD/YY
(If more than 5 values, continue on reverse or give refernce to separate sheet)		

Other editing information The date type can be changed to o ther formats like dd/mm/yy, dd/mm etc.

Related data structures / elements ______

2.9 Specimen form for recording data elements

New Emp details **DATA FLOW**

Source ref: Description : Employee

Destn. Ref: Description : Employee Register

Expanded description : Employee details like: Name, date of joinng, dept, grade, salary details, bank details are entered into the employee register

Included data structures :
Employee register

Volume Information

Volume increases as and when employee joins. Does not decrease when employee exits, as the record is not deleted. (Exit flag is inserted)

Fig 2.10 Specimen form for recording data flow

Transaction List **DATA STORE REF :**

Description : Month's Tranasction details

Data flows in :

Overtime , attendance and advance salary details

Data flows out

Consolidated transaction of the month

Contents :
in:
Employee number,
Transaction code
Transaction value

Immediate access analysis is to be found

Physical organisation : Sales Department

Fig 2.11 Specimen form for recording data store

Description : Maintain Master Employee Pay Details

Inputs	Logic Summary	Outputs
New Employee details bank details	For new employee, all required details need to be entered. A new record is written.	Updated employee master file
Current employee details to be updated	For old employees, details to be changed are updated	

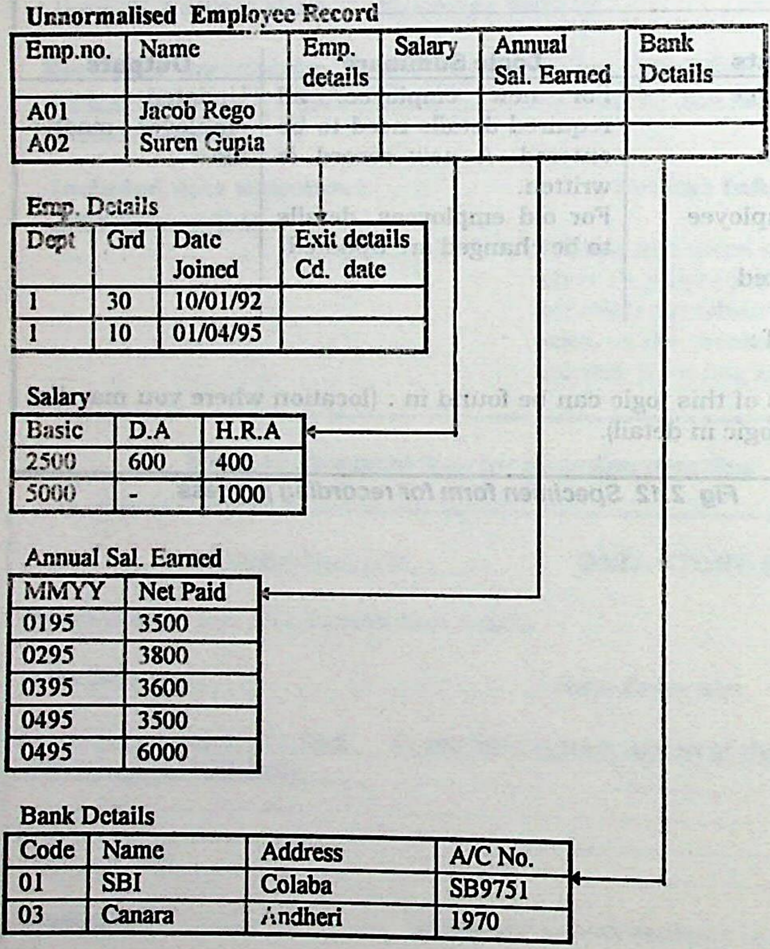
Physical ref :

Full details of this logic can be found in : (location where you may list the entire logic in detail).

Fig 2.12 Specimen form for recording process

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

Normalization



Unnormalised Employee Record

Emp.no.	Name	Emp. details	Salary	Annual Sal. Earned	Bank Details
A01	Jacob Rego				
A02	Suren Gupta				

Emp. Details

Dept	Grd	Date Joined	Exit details Cd. date
1	30	10/01/92	
1	10	01/04/95	

Salary

Basic	D.A	H.R.A
2500	600	400
5000	-	1000

Annual Sal. Earned

MMYY	Net Paid
0195	3500
0295	3800
0395	3600
0495	3500
0495	6000

Bank Details

Code	Name	Address	A/C No.
01	SBI	Colaba	SB9751
03	Canara	Andheri	1970

Figure 3.3 Unnormalised Employee Record

Figure 3.3 shows an unnormalized form of an employee record, this consists of:

Employee no., employee name, employee details (department code, grade, date of joining, exit code and exit date), annual salary earned (MMYY , net paid), bank details (bank code, bank name, address, employees A/C no) .

Here it is clearly seen that the employee's annual salary earned details which are : Month and Year paid, net paid, are being *repeated.* Therefore this relation is not a first normal form.

First Normal Form - Employee Record

Emp.no.	Namc	Emp. details	Salary	Bank Details
A01	Jacob Rcgo			
A02	Suren Gupta			

Annual Sal. Earned record

Emp.no	MMYY	Net Paid
A01	0195	3500
A01	0295	3800
A01	0395	3600
A01	0495	3500
A02	0495	6000

Fig. 3.4 First normal form

Figure 3.4 shows the normalization to first normal form for the employee record.

As mentioned above the first normal form is carried out by removing the repeating group. In this case we remove the Annual salary earned items and include them in a new file or relation called Annual Salary earned record. **Employee number is still the primary key in the employee record. A combination of employee number and MMYY is the primary key in the annual salary earned record.**

We thus form two record structures of fixed length:

Employee record consisting of: Employee no., employee name, employee details (department code, grade, date of joining, exit code and exit date), bank details (bank code, bank name, address, employees A/C no) :

Annual salary earned record consisting of - employee no., month & year(MMYY) and net paid.

Second Normal Form - Employee Record

Emp.no.	Name	Emp. details	Salary	A/C No.	Bank code
A01	Jacob Rego			SB9751	01
A02	Suren Gupta			1970	03

Annual Sal. Earned record

Emp.no	MMYY	Net Paid
A01	0195	3500
A01	0295	3800
A01	0395	3600
A01	0495	3500
A02	0495	6000

Bank Record

Code	Name	Address
01	SBI	Colaba
03	Canara	Andheri

Figure 3.5 Second Normal Form

The three record structures that are created are :

1. **Employee record** consisting of: Employee no., employee name, employee details (department code, grade, date of joining, exit code and exit date), bank details (bank code, bank name, address, employees A/C no) .

2. **Annual salary earned record** consisting of - employee no., month & year(MMYY) and net paid.

3. **Bank record** consisting of : bank code, bank name and bank address. All the attributes of this relation are ***fully dependent*** on Bank code.

The primary key of each of the record structures are underlined.

Structured English

◆ Sequence Structures

Example :
Process to update a particular employee record as he has resigned.

1. Get particular employee record
2. Enter '1' in Exit code data element
3. Enter date of resigning in exit date.
4. End of job

This simple example shows a sequence of four steps. Note that none of the steps contain a decision or any condition that determines whether the steps are taken.

◆ Decision Structures

The following example shows the nesting of multiple levels of conditions and actions for each decision point.

```
If employee does not exist
    Else
      If grade < 20
      DA = 20% of basic
      HRA = 0
    Else
       If grade < 30
       DA = 20% basic limit to 600
       HRA = 400
       Else
      If grade < 40
      DA = 0
      HRA = 40% of basic
      Else
            DA = 0
           HRA = 50% of basic salary
End if
```

◆ Iteration Structures

Example:

```
 DO WHILE there are employees
 If employee does not exist
     Else
        If grade < 20
        DA = 20% of basic
        HRA = 0
    Else
         If grade < 30
         DA = 20% basic limit to 600
         HRA = 400
         Else
       If grade < 40
       DA = 0
       HRA = 40% of basic
       Else
               DA = 0
             HRA = 50% of basic salary
End if
End do
```

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

Decision Trees

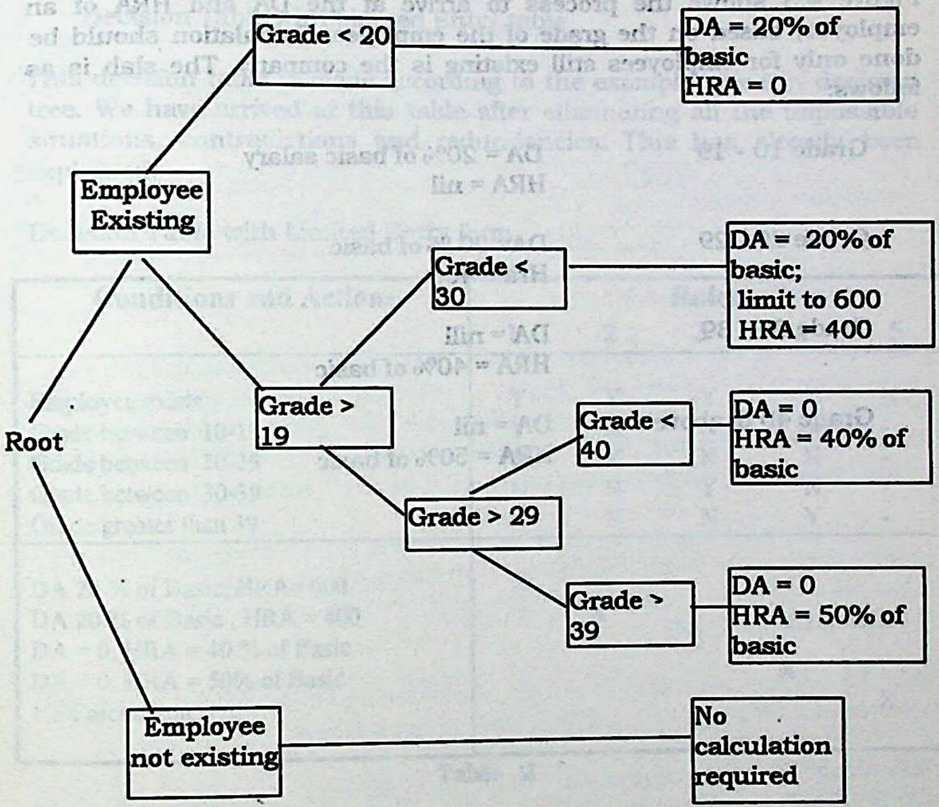


Figure 4.3

Figure 4.3 shows the process to arrive at the DA and HRA of an employee, based on the grade of the employee. Calculation should be done only for employees still existing is the company. The slab is as follows:

Grade 10 - 19	**DA = 20% of basic salary** **HRA = nil**
Grade 20 - 29	**DA= 20 % of basic** **Hra = 400**
Grade 30 - 39	**DA = nill** **HRA = 40% of basic**
Grade 40 & above	**DA = nil** **HRA = 50% of basic**

Decision Tables

Decision Table with Limited Entry table

This decision table is made according to the example given in decision tree. We have arrived at this table after eliminating all the impossible situations, contradictions and redundancies. This has already been explained.

Decision Table with Limited Entry form

Conditions and Actions	Rules				
	1	2	3	4	5
Employee exists	Y	Y	Y	Y	N
Grade between 10-19	Y	N	N	N	-
Grade between 20-29	N	Y	N	N	-
Grade between 30-39	N	N	Y	N	-
Grade greater than 39	N	N	N	Y	-
DA 20 % of Basic, HRA= 600	X				
DA 20 % of Basic , HRA = 400		X			
DA = 0, HRA = 40 % of Basic			X		
DA = 0, HRA = 50% of Basic				X	
No Calculation					X

Table 2

In a limited entry decision table, the condition are expressed as simple Yes or No questions, whereas in a Extended Entry table conditions have more than two possible states.

In the extended entry form, the condition is that, if an employee exists, and if he is in marketing department ("MKT") he is eligible for commission. The commission is based on the total sales made for the month and is calculated as follows:

If sales >=10000 then commission is 20%
if sales >=5000 and <10000 then commission is 10%
if sales <5000, commission is 0

The table is shown in table 3.

Conditions and Actions	Rules				
	1	2	3	4	5
Employee exists	Y	Y	Y	Y	N
Department	MKT	MKT	ADM	MKT	-
Total sales	10,000	2000	-	5000	-
Commission applicable	20%	0	n.a.	10%	n.a

Table 3

Note: n.a is Not Applicable

The decision table above explains how the commission is given according to the sales per month. According to these conditions the actions have to be taken. In the first rule, the person exists, he is in "MKT" department, and his total sales is upto 10000, so he is eligible for commission, he will get commission, 20% of sales amount.

In rule 3, the person is in "ADM" department, so he does not have any sales figure, so he is not eligible for any commission as such.

☞ Mixed-Entry Form

Now let us see an example of the mixed entry form, where there are various combination of conditions and actions taken.

According to the company's rules and policies, following are the conditions and actions to be taken

1) For all employee in the grade 10-19 and those in marketing department commision is calculated as follows:

If sales >=10000 then commission is 20%

if sales >=5000 and <10000 then commission is 10%

if sales <5000, commission is 0

If the person is in any other department then he is not entitled for commission.

2) **Second condition is to check if the employee is supposed to pay income tax. This is done as follows:**
If the employee's salary >= 50,000 - Tax applicable
if salary <50000 - Tax not applicable

Tax is calculated for employees of all departments.

Conditions and Actions	Rules 1	2	3	4	5
Department	MKT	MKT	FI N	FIN	MKT
Grade 10- 19	Y	Y	Y	Y	Y
Salary >= 50,000 p.a.	Y	N	Y	N	N
Salary < 50,000 p.a.	-	Y	N	Y	Y
Total sales	20000	10000	-	-	3000
Commission applicable	20%	10%	n.a	n.a	0
Income Tax applicable	X	-	X	-	-
Income Tax exempted	-	X	-	X	X

Table 4

The decision table in table 4 explains the following :

In the first rule, the person is in "MKT" department, the grade is between 10 and 19, and the salary per annum is more than 50,000 and sales is Rs. 20,000. The action taken for this rul` is - 20% commission i- given because he is in marketing department aud he is applicable for Income tax.

In rule 4, the employee is in Finance ("FIN")department, grade is between 10 -19, and salary less than 50,000 per annum, so he is exempted from income tax, and he is not given any commission as he is not in marketing department.

In rule 5, he is in marketing department, his grade is between 10-19, he need not pay income tax as salary less than 50,000, he is not given commission as his sales is only Rs. 3000.

All these tables are made while design specifications are made. Each table is made according to a required module, and you can see how each table is designed and how actions are taken.

☞ ELSE form

The conditions for the ELSE form is that the employees of marketing department are given commissions according to the total sales done.

The commission is given as follows:

If Sales >= 10000 then 20% commission
if sales >=8000 and sales<10000 then 15% commission
if sales >=5000 and sales <8000 then 10% commission
else
if sales <5000 then 2% commission

Conditions and Actions	Rules			
	1	2	3	4
Sales >= 10000	N	Y	N	E
Sales >=8000 and <10000	Y	-	N	L
Sales >=5000 and <8000	N	-	Y	S
				E
Commission	15%	20%	10%	2%

Table 5

The Else form decision table shown below has an extra ELSE column. The ELSE is applicable for all marketing department employees who have made sales less than 5000.

The first rule tells us that the employee is in marketing department, his sales amount is between 8000 and less than 10000 so he gets 15% commission.

Structure Charts

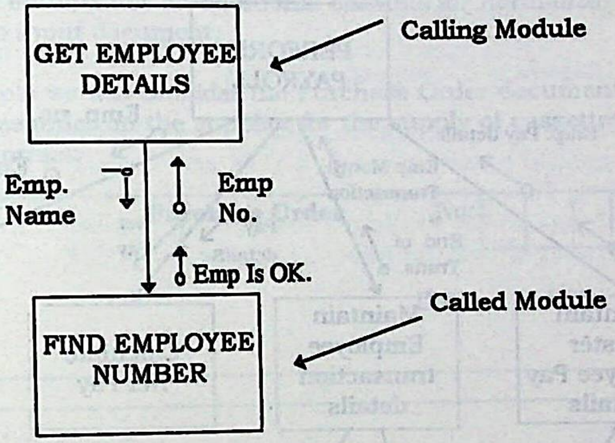


Figure 5.1 shows a structure chart

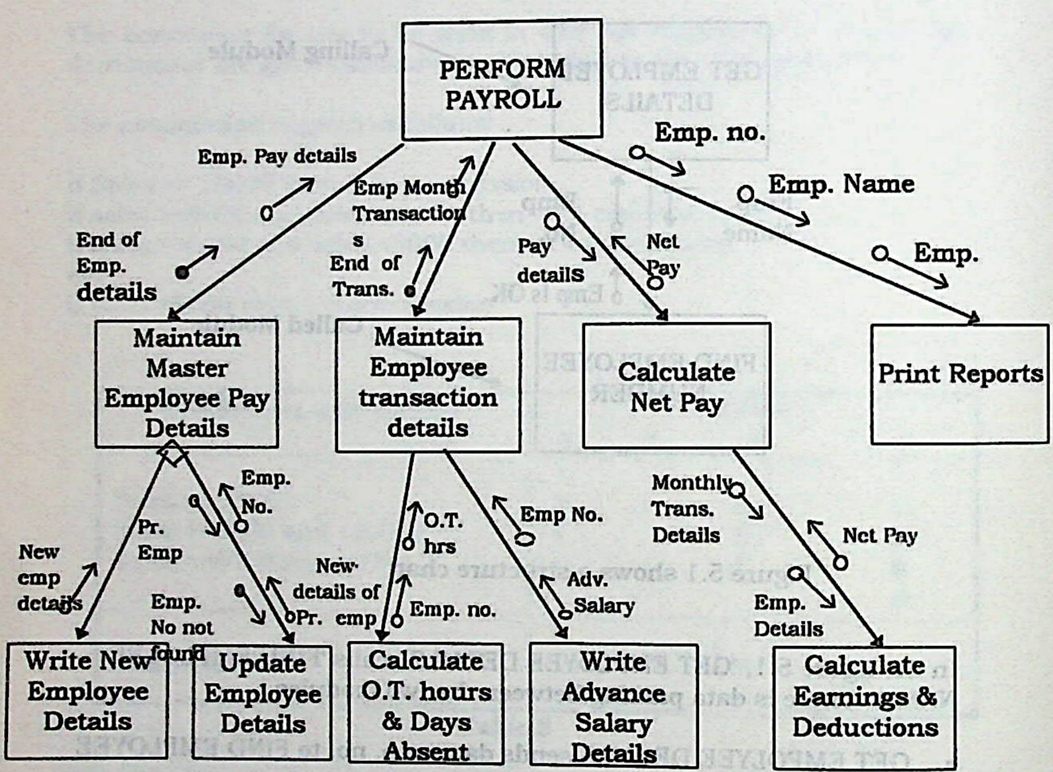
In the figure 5.1, 'GET EMPLOYEE DETAILS' calls 'FIND EMPLOYEE NAME'. There is data passing between the two modules.

- GET EMPOLYEE DETAILS sends data emp. no. to FIND EMPLOYEE NAME.

- FIND EMPLOYEE NAME (having done its function) returns data Emp. Name to GET EMPLOYEE DETAILS, and

- FIND EMPLOYEE NAME also returns a flag (Emp. No. Is OK) to GET EMPLOYEE DETAILS . This is used to tell the calling module (caller) that everything went well because sometimes GET EMPLOYEE DETAILS may send a flag saying 'invalid emp. No.'.

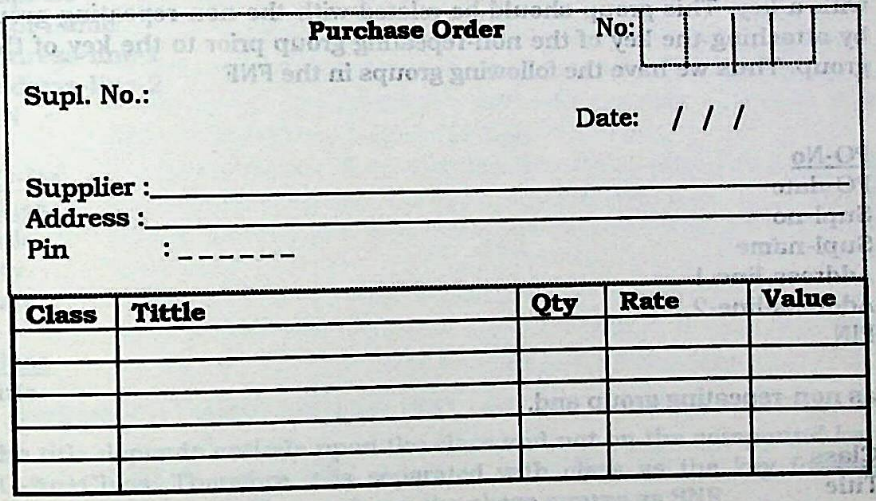
STRUCTURE CHART



Appendix B

This appendix enumerates an additional example for normalization with reference to an input document.

For this example we will consider the Purchase Order document, which is used to place order on the supplier for the supply of cassettes. It has the following format:



Purchase Order No: | | | |

Supl. No.:

Date: / / /

Supplier :________________________________

Address :________________________________

Pin : _ _ _ _ _ _

Class	Tittle	Qty	Rate	Value

The unnormalized (UNF) form has the following data items:

<u>UNF</u>

<u>PO-No</u>
PO-date
Supl-no
Supl-name
Address-line-1
Address-line-2
PIN
Class
Title
Qty
Rate

The UNF should also have a key. In the above example, PO-NO is the key. The keys are always underlined. Any derived/calculated items such as Value need not be listed.

First Normal Form (FNF)

Rule 1. Separate the repeating Group

In the above UNF, items listed after PIN are repeating. Such repeating groups should be separated from UNF and written as a separate group with a key. This group should be related with the non-repeating group by attaching the key of the non-repeating group prior to the key of this group. Thus we have the following groups in the FNF

PO-No
PO-date
Supl-no
Supl-name
Address-line-1
Address-line-2
PIN

as non-repeating group and,

Class
Title
Qty
Rate

as the repeating group.

The repeating group has Class as the key. This cannot be unique as it is repeating within the PO. Therefore if combined with PO-No which is the key of the non-repeating group then it is unique. Thus, the non-repeating group from the above two groups and the repeating group given below will form the FNF.

Po-No
Class
Title
Qty
Rate

Second Normal Form (SNF)

Rule 2. Removal of data items which are not fully dependent on the primary key.

The SNF will be as follows after applying the rule to the FNF

<u>PO-No</u>
PO-date
Supl-no
Supl-name
Address-line-1
Address-line-2
PIN

<u>PO-No</u>
<u>Class</u>
Title
Qty
Rate

<u>Class</u>
Title

The title depends entirely upon the class and not on the compound key PO-No+Class. Therefore it is separated with class as the key for the separated group. We then arrive at the above groups as SNF.

Third Normal Form

Rule 3. Removal of transitive dependencies

All non-key attributes in each group are examined to check whether there are any inter-dependencies. If found, such dependent items are separated into a different group along with the data item on which they are dependent. This data item will form the key. Hence, only the key attribute should be retained in the parent group. The first group in SNF has such dependencies. Supplier Details depend on Supplier No. There is only one supplier for a particular PO. The details of any supplier cannot be deleted from the supplier group until all the POs pertaining to that concerned supplier are deleted. Hence, every PO will have only one supplier whereas a supplier may supply items appearing in different POs.

Applying the above rule and considering transitive dependencies, we arrive at the following groups which are in TNF

TNF

PO-No
PO-Date
Supl-No

Supl-NO
Supl-Name
Address-line-1
Address-line-2
PIN

PO-NO
Class
Qty
Rate

Class
Title

All the Normal Forms (UNF thru TNF) are given below for reference

UNF

PO-No
PO-date
Supl-no
Supl-name
Address-line-1
Address-line-2
PIN
Class
Title
Rate
Qty

FNF

PO-No
PO-Date
Supl-No
Supl-Name
Address-line-1
Address-line-1
PIN

PO-No
Class
Title
Qty
Rate

SNF

PO-No
PO-Date
Supl-No
Supl-Name
Address-line-1
Address-line-2
PIN

PO-NO
Class
Qty
Rate

Class
Title

Class

TNF

PO-NO
PO-Date
Supl-No

Supl-NO
Supl-Name
Address-line-1
Address-line-2
PIN

PO-NO
Class
Qty
Rate

Title

Appendix - C

Task List For Each Life Cycle Stage

This appendix outlines in detail the tasks and deliverables for each stage of SDLC. This may be used for reference or as a check list.

Task List for the Feasibility Stage

- Analyze proposed system and write system description
- Define and document possible types of systems
- Produce cost analysis of the proposed systems. Usc data from any similar systems as a guideline.
- Produce estimate of system size, schedules and costs. Include a schedule for completion of all major deliverables.
- Define quantitative and qualitative benefits of the proposed systems.
- Produce initial pay back schedules.
- Produce a detailed estimate of costs, schedules, resources, and the like, for the next stage (requirements definition).
- Assign project manager(s).
- Produce feasibility study document .
- Present a feasibility study report to the management review committee for approval of a particular system.

Task List for the Requirements Definition Stage

- Define scope of the proposed system
  - Functions
  - Users
  - Constraints
- Interview all current and proposed users to determine the following:
  - Use of the current system
  - Deficiencies in the current system
  - Requirements of the new system
- Having determined the above, document the current system in terms of :
  - Description
  - Deficiencies

- Produce new system requirements document which should include the following :
  - Primary user
  - Requirements (process and information requirements)
  - Resolution of current system's deficiencies

- Produce list of tangible and intangible benefits.

- Produce a detailed estimate of costs, schedules, resources, and the like, for the next stage (system specification), including a schedule for production of major stages' deliverables.

- Produce a revised estimate of costs, schedules, resources, and the like, for the remainder of the project life cycle.

- Produce the requirements definition document (this task may include the building of a prototype).

- Carry out final review of requirements definition document.

- Make a decision on whether or not to continue the project.

- Define the major responsibilities of the next stage.

Task List for the System Specification Stage

- Define the type of proposed system by translating physical, environmental, and operational constraints into system characteristics (as far as possible at this stage).

- Draw up the outline of the proposed system. This includes translating user requirements from the previous stage into financial specifications.

- Develop the system dictionary by describing all elements of the system including functions, and data (information).

- Review and expand cost-benefit analysis. Update old cost-benefit analysis with new information.

- Produce a detailed estimate of costs, schedules, resources, and the like, for next stage (system design).
- Produce revised estimate of costs, schedules, resources, and the like, for the remainder of the project life cycle.
- Produce the system specification document.
- Carry out final review of system specification document
- Review the decision of whether or not to continue the project.
- Define major responsibilities for the next stage for team members, and others.

Task List for the System Design Stage

- Produce overall system design to include
  - Programs and major program functions
  - Functions and programs
  - Hardware and software environment
- Carry out a search for suitable software packages which could implement some or all of the required functionality in a cost-effective manner.
- Develop a detailed system design for each design alternative. Provide detailed system design documentation for the complete system . This will include
  - Updating the system data dictionary
  - Comparison of the design against system specification
  - Documenting the required hardware and software environment
- Update the cost-benefit analysis for each design alternative with any newfound information. Review the analysis stage findings to ensure that expected benefits still exist and that the pay back period is still acceptable.

- **Produce a detailed and revised estimate of costs, schedules, resources, and the like, for next stage (program design and development).**

- **Evaluate the design alternative and for each design alternative document the following:**
  - **User requirements being met by this alternative**
  - **Cost-benefit analysis and pay back schedules**
  - **Probable user acceptance level**
  - **Recommend the best alternative(s)**

- **Develop a test plan for the system by**
  - **Creating input test data**
  - **Preparing a list of expected output**
  - **Preparing a list of test criteria**
  - **Developing a system test schedule**

- **Identify the needs for user's training and documentation. Here, it is necessary to define an outline for :**
  - **User documentation**
  - **Operator manuals**
  - **User and operator training documents and schedules**

- **Produce the system design document.**

- **Carry out final review of system design document.**

- **Define next stage major responsibilities for programming and test team members, and others.**

Funding: Tattva Heritage Foundation,Kolkata. Digitization: eGangotri.

Task List for the Program Design and Development Stage

- Produce work plan :
  - Develop detailed list of tasks to complete development and testing of all system components
  - Produce schedules for all above tasks (a PC-based project management system could be useful for this)
  - Install progress and status recording procedures
  - Obtain approval of work plan from the project's management

- Produce detailed design for each program.

- Code, document, and unit test for each program. Carry out any and all necessary updates to the system data dictionary.

- Carry out integration test . Enter successfully unit tested programs into Integration Test Library. Carry out integration testing on each program. Document all integration test results.

- Finalize users and operators user guides and training manuals.

- Produce a detailed estimate of costs, schedules, and the like, for the next stage (system test) .

- Produce a revised estimate of costs, schedules, resources, and the like, for the remainder of the project life cycle.

- Produce program design and development document.

- Carry out review of program design and development document.

- Carry out review of system test plans.

- Obtain required sign off for completed programs.

Task List for the System Test Stage

- Following activities must be done :
  - Test system according to system test plan
  - Check out operational use of users' and operations' guides by using them to carry out the system test
  - Check out users' and operators' documents by using them to train the operators and user who carry out the system test
  - Document all system test results
- Review implementation schedule in terms of :
  - Availability of resources
  - Contingency factors that might affect implementation
  - Availability of third-party vendor support
  - Final review of detailed implementation schedule
- Develop a contingency plan which includes :
  - Criteria of contingency
  - Identification of contingency resources
  - Timetable for recovery or abandonment
- Develop service level agreement which outlines:
  - User performance, accuracy, and volume criteria
  - Vendor support criteria like Mean time to Failure and Mean time to Repair
  - System quality criteria
- Produce the system test documents
- Review and approve such documents
- Approve system documentation like :
  - Program documentation
  - Operators' manuals
  - Users' manuals
  - Support documentation

- Approve implementation plan

- Sign off fully tested system with all involved
  - System development sign off
  - Users' sign off
  - Operations sign off
  - Quality assurance sign off
  - Finance sign off

Task List for the Implementation Stage

- Install new hardware and software (this can and should be done prior to this stage, preferably during system test).

- Train first set of users and operators (this can also be done during system test stage).

- Develop contingency, recovery, and fallback plans (this can also be done during system test stage).

- Develop maintenance and release procedures and set up procedures for :
  - Regular releases of software (internal and vendor-supplied)
  - Emergency "fixes"

- Carry out any data conversion required (this may be part of the operation of the new system)

- Carry out installation of new system :
  - Immediate cut-over or
  - Parallel run or
  - Phased installation

- Plan and schedule the post implementation review and set criteria for :
  - System performance
  - System quality
  - User satisfaction
  - Quality of user and operator manuals training
  - Ensure availability of required personnel and required documentation

- **Carry out post implementation review :**
  - **Create post implementation review report**
  - **Obtain signed approval of the report**
  - **Obtain letter of system approval**

- **Set up schedules for post implementation reviews, if necessary**

Task List for the Maintenance Stage

- **Implement changes to the system.**

- **To ensure that the system continues to meet the user's needs, use the service level agreement and perform**
  - **Regular reviews of requirements of service level agreement**
  - **Regular reviews of how system is meeting those requirements**

Deliverables Required From Each Life Cycle Stage

Deliverables to be Produced from the Feasibility Stage

The feasibility study should include the following as its output :

- Brief description of the proposed system and its characteristics
- Brief description of the business need for the proposed system
- Proposed organizational structure defining key responsibilities of the project team
- Cost-benefit analysis, including a gross estimate of schedules and costs and a pay back schedule
- Proposed, tentative schedule for the delivery of key end products

Deliverables to be Produced from the Requirements Definition Stage

The output from this stage should contain the following :

- Analysis of the current system (if any)
- Set of new system user requirements
- Summary of the proposed system (this can include a prototype of the proposed system)
- Estimates of the next stage and of the remainder of the project

Deliverables to be Produced from the System Specification Stage

The output from this stage is the system specification document, which contains the following :

- System description
- Data requirement
- Network and telecommunication requirements

- System controls (password access, recovery and restart, etc.)
- Revised cost-benefit analysis and pay back schedule
- Estimates of the next stage and of the remainder of the project

Deliverables to be Produced from the System Design Stage

The output from this stage should include the following :

- Management summary of the proposed system
- Detailed system description, including descriptions and specifications of the following :
  - Programs, module libraries
  - Files and databases
  - Records and transactions
  - Data dictionary
  - Procedures
  - Schedules
  - Interfaces, both human and machine
- Description of proposed system controls
- Revised cost-benefit analysis and pay back schedule
- Recommended program design techniques, programming and documentation standards
- Recommended implementation techniques
- Preliminary system test plan
- Estimates of next stage and of the remainder of the project

Deliverables to be Produced from the Program Design and Development Stage

The output of this stage should include the following :

- Detailed design documents for the system and for each program.
- Detailed design diagrams for the system and for each program.
- Detailed logic representation for each program.
- Detailed (program) documentation for each program
- Input/Output descriptions (files, databases, transactions, screens, reports, and the like)
- Program source listings, including embedded comments.
- Operator's guide (manuals) for the complete system.
- Results of unit tests for each program.
- Results of integration tests.
- User guide (manuals) for the complete system.
- Estimates of next stages, including the implementation schedule, conversion plans, and recovery and contingency plan.
- System test plan.

Deliverables to be Produced from the System Test Stage

The outputs from the system test stage should include the following :

- System test plans (updated).
- System test results.

- Results of variance with the expected results and plans for resolution of these variances.
- Results of documentation tests i.e. whether or not each type of documentation adequately served its intended purpose.
- Implementation schedules, conversion plan, and recovery, contingency, and fallback plan
- Service level agreement

Deliverables to be Produced from the Implementation Stage

The following deliverables should be produced by the end of the implementation :

- Full set of release and maintenance procedures.
- Detailed contingency, recovery, and fallback plan (if not already produced during the previous stage).
- Schedule and plan for the post implementation review.
- Post implementation review report.
- Signed letter of system approval.
- Detailed schedule for further post implementation reviews.

Deliverables to be Produced from the Maintenance Stage

The following deliverables should be produced during the maintenance stage of the system's life cycle :

- Detailed log of changes to the system.
- Copies of regular reviews and verifications of the service level agreement.
- Copies of regular post implementation review reports.

Glossary

Analysis
Analysis means breaking a problem into successively manageable parts for individual study.

Afferent module
A module that obtains its input from its subordinates and delivers it upward to its superordinate(s).

Aliases
An alias, is an alternative name for a data element.

Alpha testing
Verifying and studying software errors and failures based on simulated users requirements.

Artificial test data
Are solely for test purposes. They are to be generated to test all combinations of formats and values.

Automated systems
These are nothing but man-made systems that interact with or are controlled by one or more computers.

Batch Processing system
In batch system, information is usually retrieved on a sequential basis, which means that the computer system reads through all records in its database, processing and updating those records for which there is some activity.

Beta testing
Subjecting modified software to the actual user site (live) environment.

Central functions
These are usually left in the middle after the afferent and efferent functions are identified. Central functions are the main work of the system. They transform the major inputs into major outputs.

Cohesion
Strength of relations **within** modules. A measure of the strength of functional association of processing activities (normally within a single module).

Constant data
This implies to data that are the same for every entry.

Coupling
It is the strength of relation **between** modules. A measure of the strength of interconnection between one module and another. The degree of dependence of one module on another, specifically, a measure of the chance that a defect in one module will appear as a defect in the other, or the chance that a change to one module will necessitate a change to the other.

Context diagram
This will be the most general diagram, really a bird's eye view of data movement in the system.

Control
In a system, the element or component that governs the pattern of activities of the system.

Conversion
Conversion is the task of translating the user's current files, forms, and databases to the format required by the new system.

Data coupling
A form of coupling in which one module communicates information to another in the form of elementary parameters.

Data Dictionary
It is an organised listing of all the data elements that are pertinent to the system, with precise, rigorous definitions so that both user and system analyst will have a common understanding of all inputs, outputs, components of stores, and intermediate calculations.

Data Elements
Data elements the most fundamental data level.

Data flow
Movement of data in a system from a point of origin to a specific destination-- indicated by a line and arrow.

Data Flow Diagram
The modelling tool that we use to describe the transformation of inputs into outputs is a data flow diagram

Data stores
Data stores could be thought of as the 'memory' of the system. In a data flow diagram, a storage area for collecting data input during processing; the symbol is a open rectangle.

Database Design
It involves designing the conceptual model of the database.

Data structure
One or more data elements in a particular relationship, usually used to describe some entity.

Decision tables
A decision table is created by listing all the relevant variables (conditions/inputs) and all relevant actions on the left side of the table.(The variables and actions have been separated by horizontal line).

Decision tree
A decision tree is a diagram that presents conditions and actions sequentially and thus shows which conditions to consider first, which second and so on.

Design
The (iterative) process of taking a logical model of a system, together with a strongly stated set of objectives for that system, and producing the specification of a physical system that will meet those objectives.

Design Specification
This is the document produced at the end of systems design.

Documentation
A thorough written description of all the component parts and operations of the system. Includes forms, personnel, equipment, and input/output sequence. Both written and charted explanation is used.

Domain
The set of values of a data element that is a part of a relation. Effectively equivalent to a field or data element.

Efferent module
A module that obtains its input from its superordinate(s) and delivers it downward to its subordinate(s).

End-users
End-user is widely used by system analysts to refer to people who are not professional information system specialists but who use computers to perform their jobs.

Exploding
Going from higher level to lower is called 'exploding' a data flow diagram.

External Entity
They are organisations, other information systems, departments or people which represent a source or destination of transactions or data.

Factored
A function or logical module is factored when it is decomposed into sub-functions or submodules.

Factoring
The separation of a function contained as code in one module into a new module of its own.

Feasibility study
An important outcome of the preliminary investigation is the determination that the system requested is feasible, which is done through feasibility study.

First normal form
A relation without repeating groups (a normalised relation) but not meeting the stiffer tests for second or third normal form.

Fixed length record
When the number and size of data item in a record are constant for every record, it is called fixed record length.

Flexibility
A measure of the degree to which a system, as is, can be used in a variety of ways.

Functionally dependent
Data item is functionally dependent if its value is uniquely associated with a specific data item.

Implementation
Includes all those activities that take place to convert from old system to the new.

Information System
Can be defined as a subsystem of the business. Specifically, it is an arrangement of interdependent human and machine components that interact to support the operational, managerial, and decision-making information needs of an organisation.

Live test data
Are those that are actually extracted from organisation files.

Logical data flow diagram
It is the model of the proposed system.

Manuals
A form of documentation to guide employees in doing their tasks.

Model
A pictorial representation of a system.

Module
A module is defined as a set of instructions which can be invoked by name. It is a group of instructions, i.e., a paragraph, block, subprogram, subroutine or the like.

Nassi-Schneiderman charts (or N-S charts)
These are graphic tools that force the designer to structure software that is both modular and top-down

Normalization
Normalization is a process of simplifying the relationship between data elements in a record. it is the transformation of complex data stores to a set of smaller, stable data structures.

On-line system
A system which accepts input directly from the area where it is created and in which the output or results of computation are returned directly to where they are required.

Physical data flow diagram
It is a model of the current system.

Process (transform)
Processes transform inputs into outputs. They are work or actions that are performed by people, machines, or computers on incoming data flows to produce outgoing data flows.

Prototype
It is a working system - not just an idea on paper - that is developed to test ideas and assumptions about the new system.

Record
A group of related fields of information treated as a unit by an application. Also called a data structure.

Record Key
To distinguish one specific record from another, one data item is selected in the record that is likely to be unique in all records of a file and use it for identification purpose.

Relation
A 'relation' is a two-dimensional table. It consists of 'rows' which represent records and columns which show the attributes of the entity. A relation is also called a file, it consists of a number of records.

Reliability
A measure of the quality of a program or system; sometimes expressed as mean-time-between-failures.

Rule
In forms design-- a rule (line) guides the human eye in reading and writing data groups and separates them on the form.

Second Normal Form
A normalised relation in which all of the nonkey domains are fully functionally dependent on the primary key.

Shared (library) modules
These are predefined procedures that are included in the system's program library. The routine is quickly invoked by a single command or call.

Software Engineering
Is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures and associated.

Source document
This is the form on which data are initially captured, i.e. recorded.

Span of control
Span of control refers to the number of sub-ordinate modules controlled by a calling module

Structured analysis
Structured analysis is a development method for the Analysis of existing, manual or automated systems, leading to the development of specifications (i.e. Design) for a new or modified system

Structure chart
A graphic tool depicting the partitioning of a system into modules, the hierarchy and organisation of those modules, and the communication interfaces between the modules.

Structured design
A set of guidelines for producing a hierarchy of logical modules which represents a highly changeable system.

Structured English
A tool for representing policies and procedures in a precise form of English using the logical structures of structured coding.

Structured flowcharts
Also called Nassi-Schneiderman charts (or N-S charts), are graphic tools that force the designer to structure software that is both modular and top-down

Structured programming
A set of guidelines and techniques for writing programs as a nested set of single entry, single-exit blocks of code, using a restricted number of constructs

Structured walkthrough
Interchange of ideas between peers who reviews a product presented by its author and agree on the validity of a proposed solution to a problem.

Subsystem
A series or group of components that perform one or more operations of a more complex system.

System
It is an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective.

System analysis
Is the process of totally understanding the current system by gathering and interpreting facts, diagnosing problems, and using the facts to improve the current system.

System Design
Detailed concentration on the technical and other specifications that will make the new system operational.

Systems development life cycle
This is a sequence of events carried out by analysts, designers and users to develop and implement an information system. These activities are carried out in different stages.

System Maintenance
Is performed most often to improve the existing software rather than to respond to a crisis or system failure.

System Testing
Testing the whole system by the user after major programs and subsystems have been tested.

Tabular format
This implies to a row-and-column format.

Testing
The critical phase of computer system development in which debugged programs are tested to ensure a working system.

Third Normal Form
A normalised -elation in which all of the nonkey domains are fully dependent on the primary key and all the nonkey domains are mutually independent.

Transaction
Any element of data, event, or change of state that causes or initiates some action or sequence of actions, usually an input.

Transaction analysis
It is the technique of identifying transaction types of a system using them as units of design.

Transform analysis
It is the strategy for converting each piece of the data flow diagram into a structure chart. Transform analysis is a 'strategy' not an algorithm.

Transitive dependency
Occurs when some of the non-key attributes are dependent not only on the primary key but also on a non-key attribute.

Tuple
A specific set of values for the domains making up a relation, it is the "relational" term for record. An individual data structure or record in a relational database.

Unit Testing
This involves the tests carried out on modules/programs which make up a system.

Validation
Checking the quality of software in both simulated and live environment.

Variable data
Those data items that change for each transaction handled or decision made.

Verification testing
Runs the system in a simulated environment using simulated data. This simulated test is sometimes called alpha testing.

Funding: Tattva Heritage Foundation, Kolkata. Digitization: eGangotri.

INDEX

APTECH
COMPUTER EDUCATION

C-95/SEM-2/05-95
Bunts /95/10/50